

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

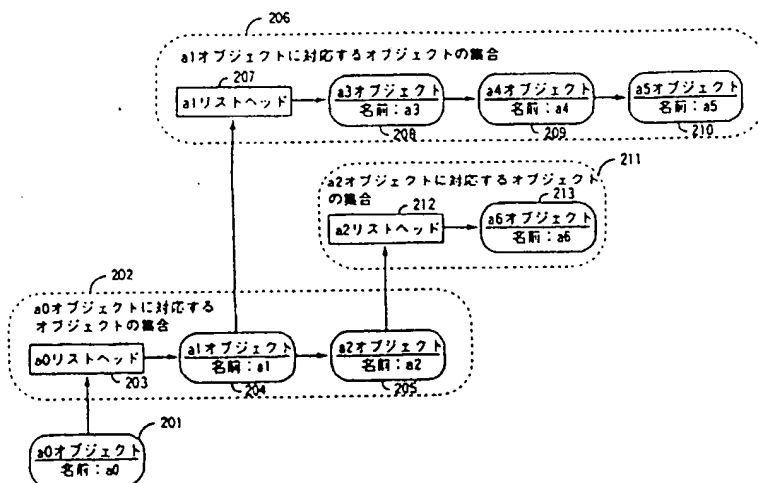
IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,
please do not report the images to the
Image Problem Mailbox.**

(51) 国際特許分類 G06F 9/44	A1	(11) 国際公開番号 WO97/29421 (43) 国際公開日 1997年8月14日(14.08.97)
(21) 国際出願番号 PCT/JP96/00227 (22) 国際出願日 1996年2月5日(05.02.96) (71) 出願人 (米国を除くすべての指定国について) 株式会社 アテナ テレコム ラボ (ATHENA TELECOM LAB, INC.)[JP/JP] 〒184 東京都小金井市東町4丁目29番15号 Tokyo, (JP) (72) 発明者; および (75) 発明者/出願人 (米国についてのみ) 上村邦夫(KAMIMURA, Kunio)[JP/JP] 〒184 東京都小金井市東町4丁目29番15号 Tokyo, (JP)		(81) 指定国 AL, AM, AT, AU, AZ, BB, BG, BR, BY, CA, CH, CN, CZ, DE, DK, EE, ES, FI, GB, GE, HU, IS, JP, KE, KG, KR, KZ, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TR, TT, UA, UG, US, UZ, VN, ARIPO特許 (KE, LS, MW, SD, SZ, UG), ユーラシア特許 (AZ, BY, KG, KZ, RU, TJ, TM), 欧州特許 (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI特許 (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG). 添付公開書類 国際調査報告書

(54) Title: METHOD AND APPARATUS FOR OBJECT MANAGEMENT

(54) 発明の名称 オブジェクト管理方法とその装置



201 ... a0 object, name: a0
 202 ... group of objects corresponding to a0 object
 203 ... a0 list head
 204 ... a1 object, name: a1
 205 ... a2 object, name: a2
 206 ... group of objects corresponding to a1 object
 207 ... a1 list head
 208 ... a3 object, name: a3
 209 ... a4 object, name: a4
 210 ... a5 object, name: a5
 211 ... group of objects corresponding to a2 object
 212 ... a2 list head
 213 ... a6 object, name: a6

(57) Abstract

A method of object management, in which object class identification information can be preserved, in object-oriented programming for future use, while it is difficult in the prior art to preserve objects and pointers. A tree-like structure (object tree) is devised for accommodating the objects, and the object identification information is given to each object. Pointer information can be preserved as a combination of the object identification information. The pointer can be reproduced by reading this pointer information and finding out the object designated by the combination of the object identification information. The object information can be distributed to a plurality of apparatuses, and a simplified virtual memory can be accomplished. High level functions for an Internet Web browser can also be accomplished by reproducing the objects and the pointer information on the basis of the information transmitted through a communication line.

(57) 要約

オブジェクト指向のプロプログラミングにおいて、従来はオブジェクトおよびポインタを保存する事は困難であった。本発明では、オブジェクトのクラスを識別するクラス識別情報を保存し、これを読み込む事によりオブジェクトの再生を可能にした。また、オブジェクトを収容するツリー状の構造（オブジェクトツリー）を考案し、それぞれのオブジェクトにオブジェクト識別情報を与える。ポインタをオブジェクト識別情報の組み合わせに変換すれば、ポインタ情報を保存する事が出来る。これを読み込み、そのオブジェクト識別情報の組み合わせで特定されるオブジェクトを見つけ出せば、ポインタが再生できる。また複数の装置にオブジェクト情報を分散させる事や、簡易化されたバーチャルメモリを実現する事もできる。また通信回線で伝えられた情報に基づきオブジェクトとポインタ情報を再生することにより、インターネットで用いるWebブラウザで高度な機能を実現する事が可能である。

情報としての用途のみ

PCTに基づいて公開される国際出版をパンフレット第一頁にPCT加盟国を同定するために使用されるコード

AL	アルバニア	EE	エストニア	LR	リベリア	RU	ロシア連邦
AM	アルメニア	ES	スペイン	LS	レソト	SD	スーダン
AT	オーストリア	FI	フィンランド	LT	リトアニア	SE	スウェーデン
AU	オーストラリア	FR	フランス	LU	ルクセンブルグ	SG	シンガポール
AZ	アゼルバイジャン	GA	ガボン	LV	ラトヴィア	SI	スロヴェニア
BB	バルバドス	GB	イギリス	MC	モナコ	SK	スロヴァキア共和国
BE	ベルギー	GE	イグルジア	MD	モルドバ	SN	セネガル
BF	ブルキナ・ファソ	GH	ガーナ	MG	マダガスカル	SZ	スワジランド
BG	ブルガリア	GN	ギニア	MK	マケドニア旧ユーゴスラ	TD	チャード
BJ	ベナン	GR	ギリシャ	UA	ウクライナ共和国	TG	トーゴ
BR	ブラジル	HU	ハンガリー	ML	マリ	TJ	タジキスタン
BY	ベラルーシ	IE	アイルランド	MN	モンゴル	TM	トルクメニスタン
CA	カナダ	IT	イタリア	MR	モーリタニア	TR	トルコ
CC	中央アフリカ共和国	JP	日本	MW	マラウイ	TT	トリニダード・トバゴ
CG	コンゴ	KE	ケニア	MX	メキシコ	UA	ウクライナ
CH	スイス	KG	キルギスタン	NE	ニジェール	UG	ウガンダ
CI	コート・ジボアール	KP	朝鮮民主主義人民共和国	NL	オランダ	US	米国
CM	カメルーン	KR	大韓民国	NO	ノルウェー	UZ	ウズベキスタン共和国
CN	中国	KZ	カザフスタン	NZ	ニュージーランド	VN	ヴェトナム
CZ	チェッコ共和国	LI	リヒテンシュタイン	PL	ポーランド	YU	ユーゴスラビア
DE	ドイツ	LK	スリランカ	PT	ポルトガル		
DK	デンマーク			RO	ルーマニア		

される。しかし、J a v a 言語ではポインタをサポートしていない。つまり、複数のオブジェクトを作成し、そのオブジェクトの間の複雑な関係をポインタで表現する高度なアプリケーションは J a v a では実現出来ない。

ポインタ情報を含むオブジェクトの情報を他の計算機に伝え、そこで再現する事が出来れば、W e b ブラウザでさらに高度な機能を実現する事が出来る。

発明の開示

〈セクション3 オブジェクトの保存と再生その1〉

オブジェクトを保存するためには、主メモリのオブジェクトの内容情報を記録媒体に書き出せば良い。この記録媒体とは、磁気ディスク、磁気テープ、フロッピーディスク、書き込み可能 C D - R O M、記憶保持型メモリなど、一時的または永続的に情報を保存する媒体の事である。主記憶媒体と対比して二次記録媒体と呼ぶ場合や外部記憶媒体と呼ぶ場合もある。また、通信回線を通じて別の計算機に伝え、その情報を保持してもらうケースもある。どちらの場合もオブジェクトの内容情報を書き出す点では同じである。

オブジェクトには、整数や実数などの値、文字列、ポインタ、等の情報が含まれている。ポインタ以外のオブジェクトの内容情報は、そのまま、記録媒体に書き出せば良い。オブジェクトの中にさらにオブジェクトが含まれる事がある。例えば、C++では、新しいクラスを定義し、そのクラスのオブジェクトを別のクラスの（オブジェクトの）変数として用いる事ができる。オブジェクトの内容情報にさらにオブジェクトが含まれる場合でも、中に含まれるオブジェクトの内容も書き出せば良い。

記録媒体に記録された情報、外部から（通信回線から、ユーザーインタフェースから）の情報からオブジェクトを再生する場合には、まずオブジェクトのひな形であるクラスから、オブジェクトを作成する。C++ならば、クラスを `class X`; とした時、`X* NewObj=new X;` でクラス X のオブジェクトが作成され、そのメモリアドレス（オブジェクトが確保された主メモリの先頭番地）が NewObj に設定される。その後、記録媒体に記録された情報を読み込み、オブジェクトの変数に値を設定すれば良い。メモリアドレスは、情報を書き出した時と異なっても、その内容は正確に再現される。

記録媒体に記録された内容を読み込む前に、オブジェクトのひな形つまりクラスが判っていれば以上の手順を実行する事が可能である。もし、保存再生するオブジェクトのクラスが1種類であれば問題無い。また、プログラムのロジックに作成するオブジェクトのクラスの指定が組込まれていれば問

題は無い。これは、例えば、記録媒体から読み込む情報のグループ毎に、作成するオブジェクトのクラスがプログラムのコードで指定されている場合である。オブジェクトの中にさらにオブジェクトが含まれる場合でも、そのクラス定義の中に、含まれるオブジェクトのクラスが明示されるので問題は無い。

保存再生するオブジェクトのクラスが複数有り、かつそのクラスの指定をプログラムのロジックに組込む事が出来ない場合に、オブジェクト保存と再生を可能にする工夫を、セクション5の「オブジェクトの保存と再生その2」で説明する。

簡単な場合のポインタの保存再生方法をセクション4で示す。より複雑なケースのポインタの保存再生方法を、セクション6とセクション11以降で示す

なお、オブジェクトの情報を記録媒体に書き出すことは、オブジェクトの情報を計算機の表示部に表示すること、通信回線を通じ外部へオブジェクトの情報を伝えること、と同じである。また記録媒体に書込まれたオブジェクトの情報からオブジェクトを再生することは、エディターで作成したファイルの情報、計算機の入力部からの情報、通信回線を通じ外部から得た情報、を用いてオブジェクトを作成することと同じである。

〈セクション4 ポインタ保存再生の問題点と、簡単な場合のポインタの保存再生方法〉

ポインタが用いられるケースを分類すると以下の3種類になる。一番目は、図1(1)に示すように、ポインタ(P1ポインタ101)とそれが示すオブジェクト(X1オブジェクト102)が1対1に対応する場合である。この場合は、P1ポインタ101に設定されたX1オブジェクト102のメモリアドレスを出力する替わりに、X1オブジェクト102の内容情報を出力すれば良い。ポインタを再生する場合は、まずオブジェクトを作成し、そのメモリアドレスをP1ポインタ101に設定する。そして、記録媒体から読み込んだX1オブジェクト102の内容を今作成したオブジェクトの変数に設定する。このようすれば、X1オブジェクト102を示すP1ポインタ101を再生できる。

二番目は、複数のポインタが一つのオブジェクトを示す場合である。図1(2)では、P2ポインタ103、Q2ポインタ104、R2ポインタ105が全てX2オブジェクト106を示している。この場合、X2オブジェクト106の内容情報を書き出し、また再生する作業をどのポインタに対応させるかが問題となる。P2ポインタ103に割当てた場合、その値つまりX2オブジェクト106のメモリアドレスをQ2ポインタ104とR2ポインタ105にも設定する必要がある。つまり、ア

アプリケーションプログラムの構造を分析し、そのアプリケーション固有の手順でポインタ情報を保存再生する処理を実現する事になる。

P 2 ポインタ 1 0 3、Q 2 ポインタ 1 0 4、R 2 ポインタ 1 0 5 が X 2 オブジェクト 1 0 6 を示す可能性はあっても、アプリケーションの処理の進展によっては、これらポインタの一部が X 2 オブジェクト 1 0 6 を示さない場合があるならば、ポインタ情報の保存再生手順はさらに困難になる。X 2 オブジェクト 1 0 6 の内容情報を書き出し、また再生する作業をどのポインタに対応させるを、状況に応じて切り替える必要がある。また、再生したオブジェクトのメモリアドレスを設定する先も、状況により変化する。また、どのポインタも X 2 オブジェクト 1 0 6 を示さない場合にも対処しなくてはならない。

三番目は、一つのポインタが複数のオブジェクトを示す可能性がある場合である。図 1 (3) では、P 3 ポインタ 1 0 7 が X 3 オブジェクト 1 0 8、Y 3 オブジェクト 1 0 9、Z 3 オブジェクト 1 1 0 を示す可能性がある事を示している。C ++ 言語では型チェックが厳密であるが、X 3 オブジェクト 1 0 8、Y 3 オブジェクト 1 0 9、Z 3 オブジェクトのクラスが異なっても、その先祖のクラスを P 3 ポインタ 1 0 7 の型とすれば、型チェックの問題は発生せず、これら 3 種類のクラスのオブジェクトを示す事が出来る。また、P 3 ポインタ 1 0 7 の型が void* 型であれば、P 3 ポインタ 1 0 7 には任意のクラスのオブジェクトのアドレスを設定する事ができる。

P 3 ポインタ 1 0 7 が示すオブジェクトの内容情報を記録媒体に書き出す事は容易であるが、それを読み込み再生する場合に、オブジェクトのクラスを特定する事が出来ない。つまり、ポインタ情報を再生する事が出来ない。

二番目と三番目をミックスした状況では問題がさらに複雑になる。

また、A オブジェクトのポインタが B オブジェクトを示し、B オブジェクトのポインタが C オブジェクトを示し、C オブジェクトのポインタが A オブジェクトを示す場合、どの順番でオブジェクトから再生するかが問題となる。

本発明のセクション 6 とセクション 1 1 以降で、上記の二番目と三番目およびこれらをミックスした状況に対応可能な、ポインタの保存再生方法を示す。これは、後で説明するようにポインタを文字列の組み合わせに変換し、記録媒体に書き出し保存する。そして、この情報を用いてポインタを再生する。

〈セクション 5 オブジェクトの保存と再生その 2〉

オブジェクトのクラスが複数有り、そのクラスの指定をプログラムのロジックに組込む事が出来ない場合は、セクション3の方法は使えない。そこで、「オブジェクトの内容情報を書き出す処理」に加え、「オブジェクトのクラス識別情報を書き出す処理」を導入する。つまり、オブジェクトの内容情報を書き出す時に、そのオブジェクトのひな形つまりクラスを特定する情報も書き出す。

オブジェクトを再現する時は、記録媒体から読み込んだクラス識別情報に従ってオブジェクトを作成し、それにオブジェクトから読み込んだオブジェクトの内容情報を設定する。これが請求項2に相当する。この他に、エディターで作成した記録媒体のファイルを読み込む場合や、通信回線を通じ外部からクラス識別情報やオブジェクトの内容情報を受信する場合や、オペレータから入力インタフェースを介して情報が入力される場合にも、請求項2の処理を用いてオブジェクトを再生（または作成）する事が出来る。

なお、この再生処理の手順を単純にするためには、情報の記録の順番は、まず、オブジェクトのクラス識別情報が記録され、引続きそのオブジェクトのオブジェクトの内容情報が記録されているのが都合がよい。しかし、順番が逆でも、読み込み側でオブジェクトの内容情報を一時待避しておけば対応可能である。また、クラスの識別情報とオブジェクトの内容情報を、（記録媒体の）別のファイルに記録し、これに対応した読み込みプログラムを作成する事も可能である。

〈セクション6 保存再生可能なポインタ〉

ポインタの値はメモリのアドレスであるので、そのまま保存しても意味がない。そこで、ポインタを保存可能な情報に変換する手順と、逆にその情報をポインタに変換する手順を考案した。これにより保存可能なポインタを実現する。これを以下に説明する。

〈セクション6.1 オブジェクトツリー〉

保存可能なポインタを実現するためには、オブジェクトの管理方法を工夫する必要がある。オブジェクトを作成した直後は、そのメモリアドレスが唯一のアクセス手順である。new コマンドでオブジェクトを作成した場合も、malloc コマンドでメモリを確保した場合も、そのコマンドのリターン値はメモリアドレスである。本発明では、そのメモリアドレスをオブジェクトの集合に収容する。その集合のメンバーをたどる事により、目的のオブジェクトにアクセスする事が出来る。

オブジェクトを収容する集合の構成法はオブジェクトにアクセス可能であれば良い。つまり、オブジェクトを収容する集合の構成法として、単純リスト、双方向リスト、アレイ、ハッシュドアレイ、

バイナリーツリー、バグ、スタック、キュー、ディクショナリー（アソシエーションの集合）など、どれで実現しても良い。ここで、ハッシュドアレイは、ハッシュ数が同じオブジェクトを収容するリストをハッシュ番号順に配列（アレイ）としたものである。なお、目的のオブジェクトを見つけ出す鍵となる情報として、後で定義する「オブジェクト識別情報」（セクション6.2参照）を用いる事が出来る。

オブジェクトの集合を管理するオブジェクトを「管理オブジェクト」とよび、管理対象の集合に対応させる。実際には、管理オブジェクトが保持するポインタ変数に、オブジェクトの集合の先頭を示すリストヘッダのアドレスを設定する。これが、請求項4（ア）に相当する。これを繰返すとオブジェクトをツリー状に収容する事が出来る。これを「オブジェクトツリー」と呼ぶ。

例を図2に示す。a0オブジェクト201にa0リストヘッダ203が対応する。これを表現するため、a0オブジェクト201が（変数として）持つポインタにa0リストヘッダ203のアドレスを設定する。以下、オブジェクトに対応するリストヘッダと言った場合、オブジェクトが持つポインタにリストヘッダのアドレスが設定されている事を意味する。a0リストヘッダ203を先頭とするリストにa1オブジェクト204とa2オブジェクト205が収容されている。これがa0オブジェクトに対応するオブジェクトの集合202である。a1オブジェクト204にa1リストヘッダ207が対応する。これを先頭とするリストにa3オブジェクト208、a4オブジェクト209と、a5オブジェクト210が収容されている。これがa1オブジェクト204に対応するオブジェクトの集合206である。a2オブジェクト205にa2リストヘッダ212が対応する。これを先頭とするリストにa3オブジェクト213が収容されている。これがa2オブジェクト205に対応するオブジェクトの集合211である。

オブジェクトツリーの最小構成は一段である。つまり、a0オブジェクト201とa0オブジェクト201に対応するオブジェクトの集合202だけで、構成されるオブジェクトツリーが最小構成の例である。

また、図2ではツリーの構成は2段であるが、必要なだけ段数を増やす事が出来る。また、a2オブジェクトに対応するオブジェクトの集合211を省略したオブジェクトツリー構成、つまりツリーの枝によって段数が異なる構成も可能である。

請求項5では「管理オブジェクトを集合のメンバーとする工程」により、複数段のオブジェクトツリーが作成可能である事を、請求の条件に入れている。これに対して、請求項4にはこの条件が無い。つまり、請求項4は1段のオブジェクトツリーしか作成できないケースも請求の対象としている。し

かし、リストを管理するオブジェクトを対応させることを請求の条件にいれているので、単にオブジェクトをリストに収容するケースは請求項4の請求対象から除外されている。

〈セクション6. 2 オブジェクト識別情報〉

オブジェクトツリーに収容するオブジェクトに、そのオブジェクトを識別する情報を設定する。これをオブジェクト識別情報と呼ぶ。文字列（例えばオブジェクトの名前）でも良いし、数字でも良い。オブジェクト識別情報として新しい変数を導入して、文字列、数字などを設定しても、また、既に定義された変数をオブジェクト識別情報として用いても良い。

オブジェクトを収容するオブジェクトの集合のなかでユニークな名前を、オブジェクト識別情報としてオブジェクトに与えると、オブジェクト識別情報の組み合わせで特定のオブジェクトを指定する事が出来る。図2の例では、オブジェクト識別情報はオブジェクトに与えられた名前（文字列）としている。（a0、a1、a5）でa5オブジェクト210を特定する事が出来る。

図2の例では、オブジェクトの名前は図2の全体のオブジェクトのなかでユニークである。しかし、オブジェクト識別情報の組み合わせでオブジェクトを特定する場合には、異なるオブジェクトの集合で同じ名前を用いる事も可能である。例えば、もし、a6オブジェクト213の名前をa5に変更しても、a5オブジェクト210は（a0、a1、a5）で、a6オブジェクト213は（a0、a2、a5）で、特定する事が出来るので、a5オブジェクト210とa6オブジェクト213が混同される事は無い。

一つのオブジェクトの集合のなかで同じオブジェクト識別情報（図2では名前）を用いても良い場合がある。この場合、オブジェクト識別情報の組み合わせで特定されるオブジェクトが複数存在する。その複数のオブジェクトを常に同じに扱うなど、アプリケーションの目的やその設計思想によっては許容される。

図2のオブジェクトツリー内のオブジェクトを指定する場合には、必ずa0からを指定することから、文字列a0の指定を省くことが出来る。例えば、a5オブジェクト210は（a1、a5）で特定する事が出来る。さらに、a0オブジェクト201自体を省略する事も可能である。但し、a0オブジェクト201に対応するa0リストヘッド203以降は残す必要がある。

本セクションで説明した、オブジェクト識別情報をオブジェクトに設定する工程が、請求項4（イ）である。

〈セクション 6. 3 ポインタの保存再生〉

オブジェクト識別情報の組み合わせでオブジェクトを特定出来ることを示した。ポインタをオブジェクト識別情報の連続に変換する工程があれば、そのオブジェクト識別情報の連続を記録媒体に書き出す工程によりポインタ情報を保存する事が出来る。これが請求項 6 である。

また、オブジェクト識別情報の組み合わせからオブジェクトを特定する工程があれば、保存されたポインタ情報をポインタに変換する事が出来る。これが、請求項 4 (ウ) である。

なお、請求項 4 および請求項 5 では、(ポインタを保存するために必要な)「ポインタをオブジェクト識別情報の連続に変換する工程」を請求の範囲にいていない。ポインタ情報を保存しなくても、記録媒体のファイルにエディターでポインタ情報としてオブジェクト識別情報の連続を書込み、これを読み込ませて、目的とするポインタを設定する使い方が可能である。また、オペレータから入力インタフェースを介して入力されたオブジェクト識別情報の連続、または通信回線から受信したオブジェクト識別情報の連続を、目的とするポインタに変換する事も可能である。

〈セクション 6. 4 効率的な探索〉

オブジェクト指向データベースでは、特定の条件を満たすオブジェクトを全て調査する場合、基本的には全てのオブジェクトをサーチする必要があるので、その処理に時間がかかる。これを避けるため、特定の条件を満たすオブジェクトの間に「関係」を設定し、調査対象のオブジェクトのグループを事前に作成する方法があるが、そのための処理が必要であり、またメモリを消費する。

一方、本発明では、オブジェクトをオブジェクトツリーに收容する。一度の調査の範囲となる可能性の高いオブジェクトをオブジェクトツリーのなかの一つの集合に集めておけば、探索は容易である。図 2 の a 1 オブジェクトに対応するオブジェクトの集合 2 0 6 に收容されたオブジェクトを調査するには、まずオブジェクト識別情報の組み合わせ (a 1) で a 1 オブジェクト 2 0 4 を特定し、対応する a 1 リストヘッド 2 0 7 を特定する。そして、a 1 リストヘッド 2 0 7 からオブジェクトを順にたどれば良い。オブジェクトを收容する仕組みがそのまま、探索の範囲を限定するために使える点が、オブジェクト指向データベースに比べ優れている点である。汎用的な使い方を想定したオブジェクト指向データベースに比べ、探索のための余分な処理やメモリを必要とする事なく効率的な探索が可能である。

一度の調査の範囲となる可能性の高いオブジェクトを一つの集合に集める事が、単純で効率の良い（高速な）探索を実現する鍵となる。多くの場合、アプリケーションの構造を分析すれば、一度の調査の範囲となる可能性の高いオブジェクトを、容易に見つける事が出来る。

例えば、通信網の模型を扱うアプリケーションでは、通信網を指定してその通信リンクをすべて順番に探索する事が多い。また、通信網のすべての通信ノードを順番に探索する事も多い。それぞれを集合としてまとめた例を図3に示す。リストヘッド301に、名前がX通信網の網オブジェクト302と、名前がY通信網の網オブジェクト303が収容されている。名前がX通信網の網オブジェクト302に対応するリストヘッド304に、名前がリンク管理の網内部管理オブジェクト305と、名前がノード管理の網内部管理オブジェクト306が収容されている。名前がリンク管理の網内部管理オブジェクト305に対応するリストヘッド311に、名前が東京<->ニューヨークのリンクオブジェクト312、名前がニューヨーク<->ロンドンのリンクオブジェクト313、名前がロンドン<->東京のリンクオブジェクト314が収容されている。名前がノード管理の網内部管理オブジェクト306に対応するリストヘッド307に、名前が東京のノードオブジェクト308、名前がニューヨークのノードオブジェクト309、名前がロンドンのノードオブジェクト310が収容されている。

X通信網のリンクオブジェクト全てを調査する場合には、（X通信網、リンク管理）でリンク管理の網管理オブジェクト305を特定し、これに対応するリストヘッド311を特定する。あとは、このリストヘッドに収容されているリンクオブジェクトを順にたどれば良い。

X通信網のリンクオブジェクト、ノードオブジェクト全てを調査する場合は、（X通信網）でX通信網の網オブジェクトを特定し、対応するリンクヘッド304に収容された全ての網管理オブジェクトについて、さらに対応するオブジェクトの集合をを調査すれば良い。

全ての通信網のリンクオブジェクトを探索する場合にはオブジェクトツリーを深さ優先探索（Depth First Search）でたどる。リンク管理の網内部管理オブジェクトのオブジェクトに到達したら、それに対応するオブジェクトの集合を調査すれば良い。ノード管理の網内部管理オブジェクトに到達した場合は、それをスキップする。

〈セクション7 オブジェクトの保存再生〉

ポイントの保存再生が困難なケースとして、セクション4では、複数のポイントが一つのオブジェクトを示す場合（図1（2））、一つのポイントが複数のオブジェクトを示す可能性がある場合（図1（3））、この両方をミックスした場合、を示した。

ところが、オブジェクトツリーを導入したことにより、このような場合でもポインタの保存再生が可能になった。したがって、オブジェクトの情報を、ポインタも含めて、完全に保存再生する事が可能になった。つまり、セクション3の「オブジェクトの保存と再生その1」、セクション4で説明した「簡単な場合のポインタ保存再生方法」、セクション5の「オブジェクトの保存と再生その2」に、セクション6の「保存再生可能なポインタ」の工夫を加える事により、オブジェクトの情報を完全に保存再生する事が可能になった。

オブジェクトの情報を保存する場合は、以上に述べた手順で情報を記録媒体のファイルに書き出す。同様の手順で、オブジェクトの情報を計算機の表示部に表示すること、通信回線を通じ外部へオブジェクトの情報を伝えること、が出来る。

オブジェクトの情報を再現する場合は、記録媒体のファイルから情報を読み込み、オブジェクトを作成する。この時、オブジェクトツリーも再生される。同様の手順で、エディターで作成したファイルの情報、計算機の入力部からの情報、通信回線を通じ外部から得たの情報、を用いてオブジェクトを作成することが出来る。

オブジェクト識別情報の連続で表現されたポインタ情報は、オブジェクトツリーが完成していないとポインタに変換出来ないケースがある。オブジェクトツリーのオブジェクトがポインタを保持している場合は、まずオブジェクトツリーを再生する。その後でオブジェクト識別情報の連続で表現されたポインタ情報を一括してポインタに変換し保存すれば、その後の処理ではポインタによる高速なアクセスが可能である。

ポインタが必要になった時点で、オブジェクト識別情報の連続をポインタに変換する方法もある。そのポインタを保存しておけば、それ以降はポインタによる高速アクセスが可能である。

〈セクション8 機能追加〉

以上の工夫をベースとして、さらに高度な機能が実現出来る。これを以下に示す。

〈セクション8.1 簡易バーチャルメモリ〉

記録媒体に記録されたオブジェクトの情報を全て主メモリに読み込むと主メモリの消費が多くなり、問題となる場合がある。これを改善するひとつの方法として、オブジェクトの集合の情報を必要になった時点で読み込む方法がある。

このために、オブジェクトの内容情報を読み込みオブジェクトを再生する工程で、他のオブジェクトを指定するポインタ情報（つまりオブジェクト識別情報の連続）を実際のポインタに変換する工程は、読み込み時点では実施しない。そのポインタが必要になった時点でオブジェクト識別情報の連続を実際のポインタへの変換を試み、その過程で（管理オブジェクトが管理する）オブジェクトの集合が主メモリに存在しないことが判明すれば、その集合を主メモリ上に展開する。オブジェクトツリーにオブジェクトを展開し、必要なオブジェクトを特定した後は、そのオブジェクトへのポインタを保持する。以降は、このポインタを使えば高速なアクセスが可能である。

主メモリの使用量が一定値以上になった場合に、使用頻度の少ないオブジェクトの集合の情報を記録媒体に書き出し、主メモリを開放すれば、簡易版のバーチャルメモリが実現できる。

なお、以上の処理のためには、オブジェクトの集合毎にその情報を記録するファイルを分けておくと都合が良い。しかし、必要無い部分を読み飛ばす処理を導入すればファイルが同じでも対応可能である。

詳細はセクション 11. 9 の「簡易バーチャルメモリの実現」で説明する。

〈セクション 8. 2 情報の分散への対応〉

計算機が通信網で接続されている場合、情報が複数の計算機に分散して保持される場合がある。情報を保持する計算機から情報を必要とする計算機に、オブジェクトの情報を送り、情報を必要とする計算機にオブジェクトのコピーを作成すれば良い。これは、セクション 8. 1 の説明での記録媒体（またはファイル）を、別の計算機に置き換えた、簡易バーチャルメモリと見る事が出来る。

情報が必要な計算機にオブジェクトの情報をコピーし、オブジェクトツリーにオブジェクトの複製を作成する。必要なオブジェクトのコピーを特定すれば、そのオブジェクトへのポインタを保持し、以降はこれにより高速なアクセスが可能である。

詳細はセクション 11. 10 の「情報の分散への対応の詳細」で説明する。

〈セクション 8. 3 Web ブラウザ〉

従来の Web ブラウザは、ネットワークから入手した各種の情報を表示するため、各種の処理プログラムが組込まれていた。一方、Java 対応の Hot Java ブラウザには、バイトコードと呼ばれる計算機の種類に依存しない実行ファイルを実行する機能が有る。例えば、特殊なフォーマットの情報を表示するための処理を Java 言語で記述し、バイトコードに変換すれば、そのバイトコード

はこのHot Javaでも動作する。Hot Javaブラウザが各種の計算機に移植されていれば、1種類のバイトコードを配付するだけで、それが各種の計算機で動作する。バイトコードでブラウザにアニメーションを表示する事も出来る。しかし、Java言語ではポインタが使えないので、実現出来る機能が制限される。

本発明により、オブジェクトとポインタ情報を通信回線により伝えることが可能になった。これをHot Javaブラウザに組込めば、サーバーから受信した情報に基づくオブジェクトの集合とポインタを、Hot Javaブラウザの管理下に再現することが出来る。このオブジェクトとポインタを操作する命令を、Java言語仕様に追加すれば、従来より複雑な処理を実現することができる。

例として、通信網の設計図を作成するアプリケーションを考える。ユーザーは、通信網に要求される条件をブラウザを介して、サーバーに伝える。サーバーでは、通信網の設計図を作成する高度で複雑な処理を、高速なCPUで実行する。その結果としての通信網の設計図は、通信網の模型として表現される。この模型では、通信ノードや通信リンクがオブジェクトで表現され、それらの関係がポインタで表現されている。

サーバーでは、本発明の方法で通信網の模型の情報を書き出し、評価手順を記述したバイトコードとともに、ユーザー側のブラウザに伝える。ユーザー側のブラウザでは通信網の模型を再現し、それを評価するバイトコードを起動する。ユーザーが、評価の指針をブラウザに伝えたと、それに従って、通信網の模型を操作する。例えば、通信ノードの一つを削除した場合の網構成を明らかにしたり、特定の通信リンクに雑音が多発した場合の影響評価などを行なう。つまり、ユーザーの観点からの各種の評価をユーザーの計算機で実行する事が可能になる。

Java言語仕様ではポインタが使用出来ないが、その理由は、システムの安全性を求めているからとされている。通信回線を通じて送受されるプログラムが、ポインタで任意のアドレスにアクセス出来るならば、重要な情報の破壊や改造が可能になる。しかし、本発明によるポインタの再現方法では、オブジェクトツリーに収容されたオブジェクトに対するポインタしか再現する事はない。従って、本発明はJavaの安全性を損う物ではない。

以上では、Hot Javaの機能拡張を前提として、本発明の利用例の一つを説明した。もしHot Javaの機能拡張が困難ならば、Hot Javaにかわるより高度な機能を提供するWebブラウザを開発する事も考えられる。

〈セクション9 請求項との対応〉

請求項2では外部から読み込んだクラス識別情報に従ってオブジェクトを作成し、それにオブジェクトから読み込んだオブジェクトの内容情報を設定する。これにより、様々なクラスのオブジェクトとその内容を、外部からの情報にもとずき作成する事が出来るようになった。

ここで、外部からの情報の例としては、記録媒体に保存された情報の他に、エディターで作成したファイルの情報、通信回線を通じ外部からの情報、入力インタフェースを介したオペレータからの情報、が有る。

請求項1は請求項2の上位概念で、「クラス識別情報を読み込みオブジェクトを作成する工程」のみを請求の条件としている。オブジェクトのクラスのみが重要な情報で、その内容を保存したり外部から指定する必要が無いケースもその請求の範囲にいられている。

請求項3は請求項2に、クラス識別情報とオブジェクトの内容情報を出力する工程を加えたものである。これにより、オブジェクトの保存と再生の両方が可能になる。

請求項4は、まず(ア)(イ)により、オブジェクトツリーを作成するための基本となる工程を指定している。これにより、ポインタ情報をオブジェクト識別情報の組み合わせで与える事が可能になった。さらに請求項4(ウ)により、オブジェクト識別情報の組み合わせが与えられれば、これをポインタに変換する事が可能になった。

なお請求項4は、請求項5の「管理オブジェクトを集合のメンバーとする工程」を指定していないので、1段のオブジェクトツリーしか作成できない場合も請求の範囲にいられている。しかし請求項4(ア)で、リストを管理するオブジェクトを対応させることを請求の条件としているので、単にオブジェクトをリストに収容するケースはその請求対象から除外している。

請求項5は請求項4に加え2段以上のオブジェクトツリーを作成する事が可能な工程を有することを請求範囲の条件としている。

請求項6は、請求項5に「ポインタをオブジェクト識別情報の連続に変換する工程」を加えた物である。つまり、ポインタを保存に適した情報に変換する工程を有することを請求範囲の条件としている。これにより、ポインタの保存と再生が可能になった。

請求項7は請求項6と請求項3を合成したもので、各種のクラスのオブジェクトの保存再生、ポインタの保存再生を可能とした。

請求項8、請求項9、請求項10、請求項11、請求項12、請求項13、請求項14は、それぞれ請求項1、請求項2、請求項3、請求項4、請求項5、請求項6、請求項7のオブジェクト管理方法をオブジェクト管理装置で表現したものである。

なお、情報の書き出しについては、記録媒体への書き出し、表示部への表示、通信回線を介した外部への通知など、その書き出し先には一切条件を付けてないない。また、情報の読み込みについても、記録媒体からの読み込み、計算機の入力部からの情報、通信回線を介した外部からの情報など、その読み込み元については一切条件を付けてないない。

図面の簡単な説明

〈セクション 10 図面の簡単な説明〉

図 1 はポインタの各種の使用例である。図 2 はオブジェクトツリーの例その 1 である。図 3 はオブジェクトツリーの例その 2（通信網の構成要素をオブジェクトツリーに収容した例）である。図 4 はオブジェクト管理方法を実現したプログラムを走行させる計算機の一般的な構成である。図 5 は「オブジェクトの作成収容工程」の詳細である。図 6 は図 7 を説明するためのオブジェクトツリーの例である。図 7 は「オブジェクトへのポインタの情報を書き出す工程」の詳細である。図 8 は「オブジェクトの情報を書き出す工程」の詳細である。図 9 は「オブジェクトの集合を書き出す工程」の詳細である。図 10 は「オブジェクトの情報を読み込む工程」の詳細である。図 11 は「最初のオブジェクトを読み込む工程」の詳細である。図 12 は「オブジェクトの集合を読み込む工程」の詳細である。図 13 は「オブジェクト識別情報の組み合わせからオブジェクトを特定する工程」の詳細である。図 14 はオブジェクト管理方法を実現したプログラムの構成例である。図 15 は図 14 のプログラムに相当する装置の実現例である。

発明を実施するための最良の形態

〈セクション 11 オブジェクト管理方法の実施形態〉

オブジェクト管理方法の一つの実施例として、計算機のプログラムとして実現する実施例がある。計算機の一般的な構成を図 4 に示す。入力部 401 から入力された指示を中央処理部 403 で分析し、指示された処理（工程）を実行する。その結果、オブジェクトが主記憶部 404 のメモリ（主メモリ）上に作成され、さらには、オブジェクトへのポインタに値が設定される。

二次記憶部 405 の記録媒体に主記憶部 404 のオブジェクトの情報、ポインタ情報が書き出される。また、表示部 402 に表示したり、通信制御部 407 を介して、別の計算機に伝えることも有る。

逆に、二次記憶部405が保持するオブジェクトの情報、ポインタ情報を中央処理部403が読み込み、これらの情報に対応するオブジェクトと、ポインタを主記憶部404に設定される。また、入力部401からの情報や、通信制御部407を経由した別の計算機からの情報を基に、オブジェクトやポインタを主記憶部に設定することもある。

〈セクション11.1 オブジェクトの作成収容工程〉

オブジェクト作成収容工程つまり、オブジェクトを作成しオブジェクトの集合に収容する工程の実施例を図5に示す。以下では図2のa2オブジェクト205を追加するケースを想定して、オブジェクトの作成収容工程の詳細を説明する。つまり、a2オブジェクト205およびa2オブジェクトに対応するオブジェクトの集合211が無い状況を想定し、そこに名前a2のオブジェクトを追加する工程を例に説明する。

まず「オブジェクトを生成する集合の指定を受け付ける」501。図2のa0オブジェクトに対応するオブジェクトの集合202を指定する場合は、オブジェクト識別情報の組み合わせ(a0)を指定する。この指定を受け付けて、a0オブジェクト201を特定し、a0リストヘッド203を特定する。このリストヘッドが、以下の工程で集合の指定として使われる。

次に「オブジェクトの名前を受け付ける」502。これは、オブジェクト識別情報として名前、つまり文字列を用いる場合である。図2のa2オブジェクト205の名前を作成する場合は、その名前「a2」を受け付ける。この他に、数字などを用いる事もできる。「指定された集合のなかでオブジェクトの名前が既に存在するかをチェック」503する工程は、リストヘッドからオブジェクトを順番のたどり指定された名前とオブジェクトの名前を照合する事により実現できる。また集合がハッシュドアレイならば、ハッシュ番号の共通するオブジェクトだけを調査の対象とする事ができる。この調査の結果、既に存在する名前ならば、「オブジェクトの名前を受け付ける」502に戻る。新規な名前である事が判明すれば、次ぎの「オブジェクトのクラスの指定を受け付ける」505に移る。作成するオブジェクトが1種類の場合、または、作成するオブジェクトの指定がプログラムのなかのコードで指定されている場合には、「オブジェクトのクラスの指定を受け付ける」505を省略する事ができる。

オブジェクトのクラスの指定をもとにオブジェクトを作成する。これが「オブジェクトを作成」506工程である。「オブジェクトの名前を受け付ける」502工程で受け付けた名前をこのオブジェクトに設定するのが「オブジェクトに名前を設定」506する工程である。図2の例ではa2オブ

ジェクト205にその名前「a2」を設定する。これが請求項4（イ）の「オブジェクト識別情報をオブジェクトに設定する工程」に相当する。

通常、オブジェクトはオブジェクト識別情報の他に整数、実数、文字列、ポインタなどの変数を持つ、また、オブジェクトの内部に別のオブジェクトを持つ場合がある。これらのオブジェクトの情報のなかには、オブジェクト作成時に設定するものがある（アプリケーションによれば無い場合もある）。これを設定するのが「その他のオブジェクトの内容情報の指定を受け付け、オブジェクトに書込む」508である。

そして、「オブジェクトを指定の集合に収容」509する。図2の例では、a0リストヘッド203につながるオブジェクトの最後に、新規に作成したオブジェクトを収容する。このオブジェクトが管理するオブジェクトの集合が無い場合はここで、オブジェクトの作成収容工程を終了しても良い。

次に「オブジェクトに対応するリンクヘッドを作成する」510。そして「オブジェクトからリンクヘッドへのポインタを設定する」511。図2の例では、a2リストヘッド212を作成し、a2オブジェクト205が保持するポインタ変数にa2リストヘッド212のメモリアドレスを設定する。これで、この先にオブジェクトを収容する準備が出来た。オブジェクト自体をリンクヘッドとする方法もあるが、セクション8.2の「情報の分散への対応」との整合性のためには、オブジェクトの外部にリンクヘッドを設けるのが望ましい。リンクヘッドから先を、別の計算機とする事が可能である。

「オブジェクトに対応するリンクヘッドを作成する」510工程と、「オブジェクトからリンクヘッドへのポインタを設定する」511工程が請求項4（ア）の「オブジェクトの集合にその管理オブジェクトを対応させる工程」に対応する。

「オブジェクトを指定の集合に収容」509する工程で収容したオブジェクトは、管理オブジェクト、つまりリンクヘッドを示すポインタを持つオブジェクト、である。したがって、この工程が、請求項5の「管理オブジェクトを集合のメンバーとする工程」に相当する。

〈セクション11.2 オブジェクトへのポインタの情報を書き出す工程〉

オブジェクトツリーに収容されたオブジェクトへのポインタの情報を書き出す工程を図7に示す。この工程の詳細を説明するために、図6を用いる。図6は図2のオブジェクトツリーの構造は同じである。個々のオブジェクトから見て、そのオブジェクトを収容する集合の管理オブジェクトを「親オ

ブジェクト」とよぶ。図6では、オブジェクトから、その親オブジェクトへのポインタが設定してある点が、図2と異なる。また、図2と区別するため、図6では名称をすべてbで初めている。

オブジェクトツリーに収容されたオブジェクトへのポインタの情報を記録媒体に書き出す必要が無い場合は、オブジェクトから管理オブジェクトへのポインタは必要ない。しかし、オブジェクトへのポインタを基に、そのポインタ情報を書き出す場合には、オブジェクトから管理オブジェクトへのポインタがあると都合が良い。以下で説明する、オブジェクトへのポインタの情報を書き出す工程では、オブジェクトから管理オブジェクトへのポインタの存在を前提としている。

「オブジェクトへのポインタの情報を書き出す工程」は、まずスタートB701から起動する。まず「ポインタの指定を受け付ける」702、そして「ポインタの先のオブジェクトを特定する」703。例として図6のb5オブジェクト610が指定されたとする。以下ではこの指定に対する処理を例として詳細を説明する。

次に「オブジェクト識別情報を書き出す」704。図6の例では、b5オブジェクト610の名前の文字列b5を出力する。この後に、別の文字列との区切り符号を出力する。改行符号、タブ符号、0コード、などを文字列の区切り符号として用いる事ができる。以下では改行符号を区切り符号とした例を示している。

次に、「親オブジェクトの指定があるかをチェック」705する。b5オブジェクトの親オブジェクトにはb1オブジェクトが指定されている。この時、「親オブジェクトを指定して“オブジェクトへのポインタの情報を書き出す工程”をスタートAから起動する」707。つまり、b1オブジェクトを指定して、同じ工程をスタートA708から、リカーシブに起動する。その結果、文字列b1が出力される。

さらに、b0オブジェクトを指定して、同じ工程をスタートAから、リカーシブに起動する。その結果、文字列b0が出力される。b0オブジェクトでは「親オブジェクトの指定があるかをチェック」705の結果がNo、つまり親オブジェクトへのポインタが示す先が無いので、処理が終わる。b5オブジェクト610を特定するオブジェクト識別情報の組み合わせは(b0、b1、b5)であるが、出力結果の文字列は「b5、改行符号、b1、改行符号、b0、改行符号」となる。

この文字列を読み込む時に、オブジェクト識別情報を組み合わせの先頭に挿入すれば、正しいオブジェクト識別情報の組み合わせが得られる。つまり、先頭から文字列を読み込む毎に、オブジェクト識別情報の組み合わせは(b5)、(b1、b5)、(b0、b1、b5)と変化する。

以上が請求項6の「ポインタをオブジェクト識別情報の連続に変換する工程」に相当する。

〈セクション 11. 3 オブジェクトの情報を書き出す工程〉

指定された一つのオブジェクトの情報を書き出す工程を図 8 に示す。

まず「書き出すファイルの指定を受け付ける」800。ここでは、ファイルのオープンが既に済んで、そのファイルポインタが与えられたとする。もし、ファイル名が与えられる場合には図 8 にさらに、ファイルのオープンとクローズの処理が必要である。

次に「対象オブジェクトの指定を受け付ける」801。例として、a1 オブジェクト 204 が指定されたとする。オブジェクトの指定からそのクラスを特定して「クラス識別情報を書き出す」802。C++ の言語使用では、オブジェクトからそのクラスを特定する情報を直接取り出す方法は示されていない。クラス定義にクラス識別情報を出力する関数を定義し、オブジェクトからその関数を呼出すのが単純かつ確実な方法である。例えば、a1 オブジェクト 204 のクラスを X とした時、そのクラスの関数として文字列 "ClassX" を出力する関数 (ClassName()) を以下の様に定義する。class X { public: void ClassName() { 文字列 "ClassX" を出力; } };。対象オブジェクトの指定が X 型のポインタ pt1 で与えられた時、「クラス識別情報を書き出す」802 工程は、pt1->ClassName(); で実行される。ここで、文字列 "ClassX" はそのクラス特有の (他のクラスのクラス識別情報と異なる) 文字列とする。

次に「オブジェクト識別情報を書き出す」803。a1 オブジェクト 204 の場合、文字列「a1」を書き出す。

以上の他にも、オブジェクトは情報を保持している。しかし、全ての情報を保存する必要は無い。アプリケーションの目的、そのプログラム構成から、保存する情報を選定する。「その他の書き出し対象のオブジェクトの情報を取り出す」804 により、その選定された情報を取り出す。整数 (int) ならば「整数を文字列に変換して書き出す」806。実数 (float) ならば「実数を文字列に変換して書き出す」807。文字列 (char*) ならば「文字列を書き出す」808。この他のデータ型として文字 (char) 倍長実数型 (double) 倍長整数 (long int) など、があるが説明を簡単にするため省略した。

「その他の書き出し対象のオブジェクトの情報を取り出す」804 によりオブジェクトツリー内部のオブジェクトへのポインタが取り出された場合は、「“オブジェクトへのポインタの情報を書き出す工程”を起動」809する。この工程はセクション 11. 2 で説明した工程であり、オブジェクトツリーに収容されたオブジェクトへのポインタのみ扱う。

図8では示していないが、それ以外のポインタでも図1（ア）の様に、ポインタとそれが示すオブジェクトが1対1の関係ならば、セクション4で述べた方法で保存可能である。つまりポインタの替わりにポインタが示すオブジェクトの情報を出力する。

「その他の書き出し対象のオブジェクトの情報を取り出す」804工程は、アプリケーションの特定のオブジェクトの特定の変数を、プログラムのコードで順番を指定して取り出す。情報の書き出し処理をクラス毎にそのメンバー関数として定義し、そのなかで、書き出し対象の情報とその順番を指定する。つまり、取り出す順番とそれを処理する手順は全てプログラムに組込まれている。図8では、これを簡略化して表現している。なお、オブジェクトの情報を読み込むプログラムでは、この順番で情報を読み込むようにプログラムが作成されている。

書き出す文字列の間には区切り符号が必要である。改行符号、タブ符号、0コード、などが文字列の区切り符号として用いる事ができる。「整数を文字列に変換して書き出す」806工程、「実数を文字列に変換して書き出す」807工程、「文字列を書き出す」808工程では、文字列を書き出したあと、区切り符号（例えば改行符号）を書き出す。

図8では示していないが、この他に、オブジェクトの中に他のオブジェクトが含まれる場合がある。C++ではクラスの定義の中に他のクラスのオブジェクトを変数として定義する事ができる。この場合は、そのオブジェクトを指定して「オブジェクトの情報を書き出す工程」を起動すれば良い。

最後に「対応する（管理する）オブジェクトの集合の情報を書き出す」811。例として図2のa1オブジェクト204が「対象オブジェクトの指定を受け付ける」801で指定されているので、a1オブジェクトに対応するオブジェクトの集合206に収容されているオブジェクトの内容を出力する。この内容の詳細は次ぎのセクション11.4の「オブジェクトの集合を書き出す工程」で説明する。

〈セクション11.4 オブジェクトの集合を書き出す工程〉

オブジェクトの集合毎にファイルを分け情報を書き出す工程を図9に示す。

セクション8.1の「簡易バーチャルメモリ」や、セクション8.2の「情報の分散への対応」では、オブジェクトの情報が、オブジェクトの集合毎にファイルを分けて記録してあると都合が良かった。図9はこれに対応したものである。

なお、セクション 11. 3 の「オブジェクトの情報を書き出す工程」の「対応する（管理する）オブジェクトの集合の情報を書き出す」811工程は、本セクションのオブジェクトの集合を書き出す工程（図9）のことである。

まず「オブジェクトの指定を受け付ける」901。図2のオブジェクトツリー全体を書き出す場合は、a0オブジェクト201を指定する。これを例として、以下で工程を説明する。オブジェクトの指定から「対応するリストヘッドを特定する」902。例ではa0リストヘッド203が特定される。

次に「ファイル名の指定を受け付ける」903。例では、ファイル名として a0.bak が指定されたとする。そして「ファイルをオープン」904する。ファイル a0.bak を開いた結果得られるファイルポインタを fp とする。

「リストヘッドからオブジェクトを順にたどり特定する」905によりオブジェクトが特定できれば、「特定したオブジェクトを指定し、またファイルを指定して“オブジェクトの情報を書き出す工程”を起動する」907。これはセクション 11. 3 で説明した工程である。ここで、ファイルの指定はファイルポインタ (fp) で渡される。

「オブジェクトの情報を書き出す工程」（セクション 11. 3）では、「対応する（管理する）オブジェクトの集合の情報を書き出す」811工程の直前までは与えられたファイルポインタ (fp) を用いて書き出す処理を行なう。

なお「対応する（管理する）オブジェクトの集合の情報を書き出す」811工程は、本セクションで説明している「オブジェクトの集合を書き出す工程」のことである。ここで問題となるのは、ファイル名の指定方法である。そこで、「ファイル名の指定を受け付ける」903工程では、「ファイルの指定が無い場合には、指定されたオブジェクトに予め与えられたファイル名を用いる」こととする。

例えば、a1オブジェクト204に対応する（管理する）オブジェクトの集合の情報を保存するファイルの名前を a1.bak とし、あらかじめ a1オブジェクト204に設定しておく。このファイル名を「オブジェクトの情報を書き出す工程」で書き出し保存する情報に加えておくと都合が良い。

また、「ファイル名の指定を受け付ける」903工程で、「ファイルの指定が無い場合には、ファイル名を自動生成する」としても良い。例えば、a1オブジェクト204の名前から、ファイル名 a1.bak を作成する。なお、ファイル名衝突が無いように注意する必要がある。

「リストヘッドからオブジェクトを順にたどり特定する」905工程で次ぎのオブジェクトが無いと判定された時、「ファイルをクローズ」908して工程は終了する。

このようにしてセクション 11. 3 の「オブジェクトの情報を書き出す工程」と、セクション 11. 4 の「オブジェクトの集合を書き出す工程」がリカーシブに交互に起動され、オブジェクトツリーが保持するオブジェクトの情報を、オブジェクトの集合毎に異なるファイルに書き出す事ができる。

オブジェクトツリーが保持する全てのオブジェクトの情報を一つのファイルに書き出す場合は、最初にオープンしたファイルのファイルポインタを次々に渡して行けば良い。

最初起動された「オブジェクトの集合を書き出す工程」でファイルをオープンしてファイルポインタを得た後、「特定したオブジェクトを指定し、またファイルを指定して“オブジェクトの情報を書き出す工程”を起動する」907工程で、セクション 11. 3 のオブジェクトの情報を書き出す工程にファイルポインタが渡される。ここまでは、ファイルが分割される場合と同じであるが、「対応する（管理する）オブジェクトの集合の情報を書き出す」811工程で、セクション 11. 4 の「オブジェクトの集合を書き出す工程」に、同じファイルポインタを渡す。つまり「オブジェクトの指定を受け付ける」901工程でそのファイルポインタを受け付ける。そして、「ファイルをオープン」904する工程と「ファイルをクローズ」908する工程は起動しない。つまり「ファイルをオープンする」904工程と、「ファイルをクローズする」904工程は、最初に起動された「オブジェクトの集合を書き出す工程」でのみ実行することになる。

このようにして、オブジェクトツリーが保持する全てのオブジェクトの情報を一つのファイルに書き出すことが出来る。

〈セクション 11. 5 オブジェクトの情報を読み込む工程〉

オープン済みのファイルポインタが与えられて一つのオブジェクトを再現する工程、を図 10 に示す。

まず「読み込むファイルの指定を受け付ける」1001。ここでファイルポインタ (fp) が与えられるとする。次に「クラス識別情報を読み込みオブジェクトを作成する」1002。セクション 11. 3 の「オブジェクトの情報を書き出す工程」で例を示したように、クラス毎にユニークな文字列（数字でも良い）がクラス識別情報である。この文字列と、クラスのオブジェクトを作成する命令との対応がわかれば、オブジェクトを作成する事ができる。予想される文字列とそれに対応する命令を求める処理の対応をプログラムに組込むのがその一つの実現方法である。

次に「オブジェクト識別情報を読み込みオブジェクトに設定する」1003。オブジェクト識別情報が文字列「a1」ならば、オブジェクトの名前にa1を設定する。つまりa1オブジェクト204が再度出現する。

オブジェクト識別情報の他に、そのクラスのオブジェクトとして再現する必要がある情報がある。セクション11.3の「オブジェクト情報を書き出す工程」で述べた様に、情報の書き出し処理はクラス毎にそのメンバー関数として定義され、そのなかで、書き出し対象の情報とその順番が指定されている。情報の読み込み処理もクラス毎にそのメンバー関数として定義され、そのなかで、書き出しと同じ対象を同じ順番で指定する。「その他のオブジェクト情報を読み込む」1004工程では、その順番に従って情報を読み込み対応する変数に設定する。つまり読み込みの順番は、情報の書き出し処理の順番に対応させる。なお、図10ではこれを簡略化して表現している。

整数の変数を設定する場合は、文字列を読み込み「文字列を整数に変換してオブジェクトの変数に設定する」1005。実数の変数を設定する場合は、文字列を読み込み「文字列を実数に変換してオブジェクトの変数に設定する」1006。文字列(char*)の変数を設定する場合は、文字列を読み込み「文字列をオブジェクトの変数に設定する」1007。この他のデータ型として文字(char)倍長実数型(double)倍長整数(long int)など、があるが説明を簡単にするため省略した。

オブジェクトツリー内のオブジェクトを示すポインタの情報は、オブジェクト識別情報の組み合わせ(文字列の連続)で読み込まれる。この場合は「文字列の組み合わせをポインタ展開情報としてオブジェクトに設定する」1008。つまりこの連続した文字列をそのまま読み込み、ポインタへの変換は実際にポインタを使用する時に行なう。

たとえば、図2のa1オブジェクト204にポインタ変数pptrがあり、a5オブジェクト210のアドレスが設定されていたとする。セクション11.4の「オブジェクトの集合を書き出す工程」により、pptrの情報は連続した文字列「a1」「a5」に変換されファイルに書き出される。さて、a1オブジェクト204を再現する過程でa5オブジェクト210へのポインタ情報を読み込むと連続した文字列「a1」「a5」が得られる。この時、ポインタ変数にはゼロを設定する(pptr=0;)。そしてポインタpptrに対応するポインタ展開情報として、文字列の組み合わせ(a1、a5)を保存する。ポインタが必要になってから、(a1、a5)用いてa5オブジェクト210をオブジェクトツリーの中で特定して、そのメモリアドレスをpptrに設定する処理は、セクション11.8の「オブジェクト識別情報の組み合わせからオブジェクトを特定する工程」で説明する。

図10では示していないが、それ以外のポインタでも図1（ア）の様に、ポインタとそれが示すオブジェクトが1対1の関係ならば、セクション4で述べた方法で再生可能である。つまりポインタが示すべきオブジェクトを再生し、そのメモリアドレスをポインタに設定する。

この他に、図10では示していないが、オブジェクトの中に他のオブジェクトが含まれる場合がある。C++ではクラスの定義の中に他のクラスのオブジェクトを変数として定義することができる。この場合は、そのオブジェクトを指定して、本セクションの「オブジェクトの情報を読み込む工程」を起動すれば良い。

オブジェクトの中のオブジェクトは、C++ではクラス定義の中に、別のクラスのオブジェクトを変数として記述する事により定義される。そして外側のオブジェクトを作成する時に同時に内部のオブジェクトも作成される。従って、オブジェクトの中のオブジェクトを読み込む場合は「クラス識別情報を読み込みオブジェクトを作成する」1002工程は不要である。

指定された変数全てを指定された順番で読み込むと、「対応する（管理する）オブジェクトの集合の情報を読み込む」1010工程に移る。

しかし、オブジェクトツリーが保持するオブジェクトの情報が、オブジェクトの集合毎に異なるファイルに書き出されており、かつ必要なオブジェクトの集合を必要な時点で読み込むことを意図する場合は、「対応する（管理する）オブジェクトの集合の情報を読み込む」1010工程では単に、管理オブジェクトからリストヘッドへのポインターをゼロに設定する。セクション11.8の「オブジェクト識別情報の組み合わせからオブジェクトを特定する工程」で、必要なオブジェクトの情報が読み込まれる。

オブジェクトツリーの全てのオブジェクトの情報を一度に読み込む場合は、「対応する（管理する）オブジェクトの集合の情報を読み込む」1010では、管理オブジェクトからリストヘッドへのポインターをゼロに設定する他に、実行する処理が有る。その詳細はセクション11.11の「オブジェクトツリーの全体を読み込む場合」で説明する。

最後に「作成したオブジェクトのメモリアドレスを報告する」1011。これは「クラス識別情報を読み込みオブジェクトを作成する」1002工程で作成したオブジェクトのメモリアドレスを起動元に報告する。

「クラス識別情報を読み込みオブジェクトを作成する」1002工程が、請求項1に相当する。
「オブジェクト識別情報を読み込みオブジェクトに設定する」1003工程、「その他のオブジェク

ト情報を読み込む」1004工程、「文字列を整数に変換してオブジェクトの変数に設定する」1005工程、「文字列を実数に変換してオブジェクトの変数に設定する」1006工程、「文字列をオブジェクトの変数に設定する」1007工程、「文字列の組み合わせをポインタ展開情報としてオブジェクトに設定する」1008工程、ポインタと1対1に対応するオブジェクトの情報を読み込む工程、および、オブジェクトの中のオブジェクトの情報を読み込む工程、などが請求項2に相当する。

《セクション11.6 最初のオブジェクトを読み込む工程》

最初のオブジェクトの情報を読み込む処理を図11に示す。まず「読み込むファイルの指定を受け付ける」1101。「ファイルをオープンする」1102工程の後、「“オブジェクトの情報を読み込む工程”を起動する」1103。これはセクション11.5の工程である。最後に「ファイルをクローズする」1104。

ここでは、オブジェクトツリーが保持するオブジェクトの情報が、オブジェクトの集合毎に異なるファイルに書き出されており、必要なオブジェクトの集合を必要な時点で読み込むことを意図する場合（セクション8.1参照）を想定する。この場合、「対応する（管理する）オブジェクトの集合の情報を読み込む」1010工程ではなにもしない。

つまり、オブジェクトツリーの最初のオブジェクトの情報がファイルより再現される。図2の例ではa0オブジェクト201が再現される。これ以外のオブジェクトはポインタを展開する工程、つまりセクション11.8の「オブジェクト識別情報の組み合わせからオブジェクトを特定する工程」のなかで読み込まれる。

オブジェクトツリーが保持する全てのオブジェクトの情報を一度に読み込む場合の詳細はセクション11.11の「オブジェクトツリーの全体を読み込む場合」で説明する。

《セクション11.7 オブジェクトの集合を読み込む工程》

「オブジェクトの集合を読み込む工程」の詳細を図12に示す。

まず「オブジェクトの指定を受け付ける」1201。ここで、図2のa0オブジェクト201が指定されたとする。「対応するリストヘッドを作成」1202し、そのアドレス（つまりa0リストヘッド203のアドレス）を、a0オブジェクト201が持つリストヘッドへのポインターに設定する。

次に「対応する（管理する）オブジェクトの集合の情報が書き出されたファイル名を特定する」1203。セクション11.4で説明したように、このファイル名の指定方法にはいくつかあった。ここでは、指定されたオブジェクトの名前からファイル名を自動生成する方法を仮定し処理を説明する。a0オブジェクト201が指定されているので、a0オブジェクト201のオブジェクト識別情報（文字列a0）を用いて、a0.bak をファイル名として特定する。なお、この名前は、セクション11.4の「オブジェクトの集合を書き出す工程」で用いるファイル名と一致する必要がある。

セクション11.4の「オブジェクトの集合を書き出す工程」で管理オブジェクトに対して、ファイル名が外部から与えられた場合には、ファイル名をその管理オブジェクトで保存する情報とし、セクション11.5の「オブジェクトの情報を読み込む工程」で読み込み再現する必要がある。

特定された「ファイルをオープン」1204し、「オブジェクトの情報を読み込む工程」を起動し、報告されたオブジェクトをリストに収容する」1205。「ファイルに情報がまだあるかをチェック」1206し、あれば「オブジェクトの情報を読み込む工程」を起動し、報告されたオブジェクトのメモリアドレスをリストに収容する」1205工程を繰返す。これにより、a0オブジェクト201に対応するオブジェクトの集合202が再構成される。「ファイルに情報がまだあるかをチェック」1206する工程の結果、もうないことが判明すれば、「ファイルをクローズ」1207して終了する。

なおここでは、オブジェクトの集合毎にファイルが異なり、そのファイル毎に情報を読み込むケースを想定している。

《セクション11.8 オブジェクト識別情報の組み合わせからオブジェクトを特定する工程》

セクション11.5の「オブジェクトの情報を読み込む工程」で説明したように、ポインタ情報は、ポインタの示すオブジェクトをオブジェクトツリー内で特定するためのオブジェクト識別情報の組み合わせ（文字列の連続）として読み込まれる。

たとえば、図2のa1オブジェクト204からa5オブジェクト210へのポインタ変数にはゼロが設定されている（pptr=0;）が、ポインタ pptr に対応するポインタ展開情報として、文字列の組み合わせ（a1、a5）が指定されているとする。また、ユーザーインタフェースからポインタ情報から直接、文字列a1と文字列a5が指定される場合もある。

ポインタ ppri を使用する前に、pptr の値をチェックする。pptr の値がゼロである場合は、対応するポインタ展開情報を参照し、（a1、a5）を得る。これからポインタつまりa5オブジェクト

210のメモリアドレスを求める手順を図13に示す。これが、「オブジェクト識別情報の組み合わせからオブジェクトを特定する工程」である。

以下では図2のオブジェクトツリーのうち、a0オブジェクト201のみ読み込まれた状態で、オブジェクト識別情報の組み合わせとして(a1、a5)が指定されたケースを想定し、これからオブジェクトを特定する工程の詳細を説明する。

まず「探索開始オブジェクトの指定を受け付ける」1301。通常、オブジェクトツリーの根元を指定する。図2のオブジェクトツリーの場合はa0オブジェクト201を指定する。なお、探索開始オブジェクトの指定が無い場合は、オブジェクトツリーの根元のオブジェクトが指定されたと解釈して工程を進める方法も可能である。

次に「オブジェクト識別情報の組み合わせの指定を受け付ける」1302。ここで(a1、a5)が指定される。もし、オブジェクト識別情報の組み合わせとして(a0、a1、a5)が指定されると先頭のa0が探索開始オブジェクトの指定に相当する。ここでは説明を単純にするため、探索開始オブジェクトをa0オブジェクト201、オブジェクト識別情報の組み合わせとして(a1、a5)が指定されたとする。

次に「指定されたオブジェクトのリストヘッドへのポインターをチェックする」1303。ポインターの値がゼロならば、対応するオブジェクトの集合の情報がまだ読み込まれていない(セクション11.5参照)ので、「オブジェクトの集合を読み込む工程を起動する」1305。この工程はセクション11.7で説明した。これにより「a0オブジェクト201に対応するオブジェクトの集合」202が再生される。

次に「オブジェクト識別情報の組み合わせの先頭のオブジェクト識別情報を持つオブジェクトを、探索開始オブジェクトに対応するオブジェクトの集合のなかから探す」1306。例では、a0リストヘッド203につながるオブジェクトのなかから、名前a1のオブジェクトを探す。そして「オブジェクト識別情報の組み合わせをコピーし、先頭のオブジェクト識別情報を削除する」1307。例では、(a5)が得られる。関数の引数として渡される値は通常コピーされるが、オブジェクト識別情報の組み合わせへのポインターで情報が渡された場合は、その情報を加工する前に、コピーが必要である。

「コピーしたオブジェクト識別情報に中身があるかを判定」1308する。例では中身があるので、「探し出したオブジェクトを探索開始オブジェクトとし、コピーしたオブジェクト識別情報の組み合わせを指定のオブジェクト識別情報の組み合わせとし、オブジェクト識別情報の組み合わせからオブ

ジェクトを特定する工程”を再度起動する」1310。つまり、縮小された条件で、今説明している関数をリカーシブに起動する。

次ぎの起動では、探索開始オブジェクトをa1オブジェクト204、オブジェクト識別情報の組み合わせとして(a5)が指定される。「オブジェクト識別情報の組み合わせの先頭のオブジェクト識別情報を持つオブジェクトを、探索開始オブジェクトに対応するオブジェクトの集合のなかから探す」1306結果、a5オブジェクト210が探し出される。また、コピーし先頭のオブジェクト識別情報を削除すると、オブジェクト識別情報の組み合わせの内容が無くなる。つまり「コピーしたオブジェクト識別情報に中身があるかを判定」1308の結果がNoなので、「探し出したオブジェクトのメモリアドレスを記録し報告する」1311工程に直接進む。

「探し出したオブジェクトのメモリアドレスを報告する」1311工程では起動元にそのメモリアドレスを報告する。これがオブジェクト識別情報の組み合わせからオブジェクトを特定する工程の結果である。なお、そのメモリアドレスを、起動元で記録しておけば、次のアクセス時にはそれを使い高速なアクセスが可能である。

以上が請求項4(ウ)「オブジェクト識別情報の組み合わせからオブジェクトを特定する工程」に相当する。

〈セクション11.9 簡易バーチャルメモリの実現〉

セクション11.8の「オブジェクト識別情報の組み合わせからオブジェクトを特定する工程」により、メモリ効率と処理の高速性を満たすアプリケーションが実現できる。

初めは、オブジェクトツリーの根元のオブジェクト(図2ではa0オブジェクト201)のみを主メモリ上に再現し、ポインタでオブジェクトにアクセスする段階で、必要なオブジェクトの集合を主メモリに再現する。一度ポインタに変換すれば、次ぎからはそのポインタを使い高速な処理が実現出来る。再現するオブジェクトの集合が増えて、主メモリが不足した場合でも、OSのメモリ管理機能があれば、アプリケーションでなにもなくても良い。

しかし、よりきめこまかに主メモリを管理する場合は、アプリケーションでメモリ管理を行なう。セクション8.1の「簡易バーチャルメモリ」で紹介した方法は、主メモリの使用量が一定値以上になった場合に、使用頻度の少ないオブジェクトの集合の情報を記録媒体に書き出し、主メモリを開放する方法である。

オブジェクト識別情報の組み合わせがポインタに変換されてその値が保持してあっても、その先のオブジェクトが開放されていれば、そのポインタは使えない。これを解決する方法を以下に示す。

まず、従来のポインタに相当する情報を（ポインタ、オブジェクト識別情報の組み合わせ、タイムスタンプ記録変数）の組み合わせで表現する。ポインタの初期値はゼロ、タイムスタンプ記録変数の初期値はゼロとする。

また、共通変数として「開放時刻共通変数」を定義し初期値をゼロとしておく。

オブジェクト識別情報の組み合わせが与えられれば、セクション 11. 8 の「オブジェクト識別情報の組み合わせからオブジェクトを特定する工程」により、ポインタに値を設定することが出来る。先にポインタの値が与えられた場合でも、それからオブジェクト識別情報の組み合わせを求める事が出来る。この工程はセクション 11. 2 の「オブジェクトへのポインタの情報を書き出す工程」で書き出す先を主メモリに変更した工程で実現できる。

ポインタを使う時には、まず開放時刻共通変数をチェックする。その値が 0 で、目的のポインタの値が設定してあれば、そのポインタを用いる。

主メモリの節約のため、参照しているポインタがあるにも関わらず、オブジェクトの集合を開放した場合には、その開放時刻を開放時刻共通変数に設定する。

再度ポインタを用いる時に、開放時刻共通変数に値が有る事が判明する。この時は、ポインタに対応するタイムスタンプ記録変数の値がゼロか、または開放時刻共通変数の時刻よりも、ポインタに対応するタイムスタンプ記録変数の時刻が前ならば、改めて、オブジェクト識別情報の組み合わせをポインタに変換し、その時刻をタイムスタンプ記録変数に記録する。

この仕組みにより、オブジェクトが開放される前に、オブジェクト識別情報の組み合わせを変換し設定したポインタは、オブジェクトの開放以降に再度設定される。

主メモリの使用量が一定値以上になった場合には、オブジェクトの使用状況から、特定のオブジェクトを指定して、セクション 11. 4 の「オブジェクトの集合を書き出す工程」を起動する。例として図 2 の a 2 オブジェクト 2 0 5 が指定されたとする。また、ファイル名 a2.bak は a 2 オブジェクト 2 0 5 に事前に指定されているか、または a 2 オブジェクト 2 0 5 の名前 a 2 から自動的に作成される事とする。

この後、セクション 11. 3 の「オブジェクトの情報を書き出す工程」と、セクション 11. 4 の「オブジェクトの集合を書き出す工程」がリカーシブに起動されるので、初めに指定したオブジェクト以降のオブジェクトツリーの内容が、オブジェクトの集合毎に異なるファイルに書き出される。図

2 の例では、a 2 オブジェクトに対応するオブジェクトの集合 2 1 1 のオブジェクトの情報がファイル a2.bak に出力される。

そして、ファイルに情報が書き出されたオブジェクトを主メモリから開放する。例では、a 2 オブジェクトに対応するオブジェクトの集合 2 1 1 を主メモリから開放する。最後に、a 2 オブジェクト 2 0 5 が保持する a 2 リストヘッド 2 1 2 へのポインタの値をゼロにする。これが、セクション 1 1. 8 の「オブジェクト識別情報の組み合わせからオブジェクトを特定する工程」のなかで、「オブジェクトの集合を読み込む工程を起動する」1 3 0 5 工程のきっかけとなる。つまり、開放されたオブジェクトの情報を再度読み込み再生するきっかけとなる。

このようにして簡易バーチャルメモリが実現出来る。

オブジェクト識別情報の組み合わせからオブジェクトを特定するために、行なうべき作業は多い。これを簡単に使えるようにするためには、新しいクラス（ポインタクラス）`class Ptr` を定義すると良い。このクラスが保持する変数としてポインタ、オブジェクト識別情報の組み合わせを収容するリスト、タイムスタンプ記録変数、を定義する。また、関数として、`void* Ptr::operator->()` | "目的のオブジェクトのアドレスを探索し報告する処理" | を定義する。この処理の内容はセクション 1 1. 8 と本セクションで説明した。`Ptr::operator->()` の戻り値を目的のオブジェクトの型にするには、Safe Casting などの手法を用いて型変換を行えば良い。

なお、参照しているポインタがあるにも関わらず、オブジェクトの集合を開放することはしない事とし、処理を単純にする事も可能である。つまり、ポインタの値があればその値を報告し、ポインタの値がゼロならば、オブジェクト識別情報の組み合わせからオブジェクトを特定し、そのメモリアドレスをポインタに設定し、呼出元に報告する。

クラス `Ptr` には、ポインタ情報を書き出す関数 `void Ptr::PtrOut()` | "オブジェクト識別情報の組み合わせを収容するリストの内容があればそれを出力する、なければポインタの内容をオブジェクト識別情報の連続に変換して出力する処理" | を定義する事ができる。また、ポインタ情報を読み込む関数 `void Ptr::PtrIn()` | "オブジェクト識別情報を読み込みオブジェクト識別情報の組み合わせを収容するリストに収容する処理" | も定義することが出来る。

宣言 `Ptr APtr`; でポインタクラスの変数が定義された場合、`APtr.PtrIn()`; でオブジェクト識別情報の組み合わせが読み込まれる。`APtr.operator->()`; で目的のオブジェクトのメモリアドレスを特定する事が出来る。そして `APtr.PtrOut()`; でオブジェクト識別情報の組み合わせが出力される。

〈セクション 11. 10 情報の分散への対応の詳細〉

セクション 8. 2 で述べたように「情報の分散への対応」は、記録媒体を別の計算機に置き換えた、簡易バーチャルメモリとして実現出来る。そのために、まず、リストヘッドを示す管理オブジェクトのポインタに加え、計算機を指定する変数を導入する。

例として、図 2 のオブジェクトの情報の中で、a 1 オブジェクト 204 に対応するオブジェクトの集合 206 のみが P 計算機に収容されているとする。それ以外の情報とそれにアクセスする処理は Q 計算機にあるとする。この時、a 1 オブジェクト 204 の、計算機を指定する変数は P 計算機に設定されている。また、a 1 オブジェクト 204 のリストヘッドへのポインタはゼロが設定されているとする。

オブジェクト識別情報の組み合わせとして (a 1、a 5)、探索開始オブジェクトとして a 0 オブジェクト 201 が指定されたとする。そして「オブジェクト識別情報の組み合わせからオブジェクトを特定する工程」で a 1 オブジェクト 204 のリストヘッドへのポインタがゼロで有る事を検出する。

この時、a 1 オブジェクト 204 の、計算機を指定する変数を参照し、P 計算機にオブジェクトの集合の情報を要求する。つまり、a 1 オブジェクトが管理する情報を転送するように、P 計算機に要求する。P 計算機からの情報を読み込み、オブジェクトの集合のコピーを作成する手順は、セクション 11. 7 の「オブジェクトの集合を読み込む工程」である。つまり、セクション 11. 8 の「オブジェクト識別情報の組み合わせからオブジェクトを特定する工程」との違いは、情報の提供元がファイル（記録媒体）であるか、計算機であるかの違いである。セクション 11. 9 の「簡易バーチャルメモリの実現」で述べた方法も、情報の提供元をファイル（記録媒体）から計算機にかえれば適用可能である。

情報が必要な計算機にオブジェクトの情報をコピーし、オブジェクトツリーにオブジェクトの複製を作成し、必要なオブジェクトのコピーを特定すれば、そのオブジェクトへのポインタを保持する。以降、これを使えば高速なアクセスが可能である。

オブジェクトの情報の変更が、複数のプロセスで行なわれる場合は、常に最新の情報を得る工夫が必要である。その一つとして、ポインタが必要な場合に、毎回、オブジェクト識別情報の連続をポインタに変換する方法がある。

〈セクション 11. 11 オブジェクトツリーの全体を読み込む場合〉

セクション 11. 5 の「オブジェクトの情報を読み込む工程」で説明したように、オブジェクトの集合毎に情報を読み込む場合には、「対応する（管理する）オブジェクトの集合の情報を読み込む」1010工程は、単に、管理オブジェクトからリストヘッドへのポインターをゼロに設定するのみであり、この他にはなにもしない。

しかし、複数のファイルに分割して出力されている情報を一括して読み込む場合や、オブジェクトツリー全体の情報を一つに出力されている情報を一括して読み込む場合には「対応する（管理する）オブジェクトの集合の情報を読み込む」1010工程でさらに実行する処理がある。

〈セクション 11. 11. 1 複数のファイルの場合〉

オブジェクトの集合毎に分割された複数のファイルからオブジェクトツリー全体を読み込む処理を、その流れにそって説明する。まず、セクション 11. 6 の「最初のオブジェクトを読み込む工程」を起動する。「オブジェクトの情報を読み込む工程を起動する」1103工程により、セクション 11. 5 の「オブジェクトの情報を読み込む工程」が起動される。ここで、「対応する（管理する）オブジェクトの集合の情報を読み込む」1010工程で、セクション 11. 7 の「オブジェクトの集合を読み込む工程」を起動する。

ここで、オブジェクトの指定を受け付ける」1201での指定は、セクション 11. 5 の「オブジェクトの情報を読み込む工程」の「クラス識別情報を読み込みオブジェクトを作成する」1002工程で作成したオブジェクトである。

以下、セクション 11. 5 の「オブジェクトの情報を読み込む工程」とセクション 11. 7 の「オブジェクトの集合を読み込む工程」が交互にリカーシブに起動されることにより、オブジェクトツリー全体の情報が読み込まれる。

〈セクション 11. 11. 2 一つのファイルの場合〉

一つのファイルからオブジェクトツリー全体を読み込む処理を、その処理の流れにそって説明する。まず、セクション 11. 6 の「最初のオブジェクトを読み込む工程」を起動する。「オブジェクトの情報を読み込む工程を起動する」1103工程により、セクション 11. 5 の「オブジェクトの情報を読み込む工程」が起動される。ここで、「対応する（管理する）オブジェクトの集合の情報を読み込む」1010工程で、セクション 11. 7 の「オブジェクトの集合を読み込む工程」を起動する。

なお、オブジェクトの指定を受け付ける」1201での指定は、セクション11.5の「オブジェクトの情報を読み込む工程」の「クラス識別情報を読み込みオブジェクトを作成する」1002工程で作成したオブジェクトである。

さらに、「ファイルをオープンする」1102工程の結果得られたファイルポインターも、セクション11.5の「オブジェクトの情報を読み込む工程」を経由して、セクション11.7の「オブジェクトの集合を読み込む工程」に渡される。このファイルポインターから情報を読み出すので、「対応する（管理する）オブジェクトの集合の情報が書き出されたファイル名を特定する」1203工程、「ファイルをオープン」1204する工程、「ファイルをクローズ」1207する工程はスキップされる。

この様にして、セクション11.6の「最初のオブジェクトを読み込む工程」を起動することにより、セクション11.5の「オブジェクトの情報を読み込む工程」とセクション11.7の「オブジェクトの集合を読み込む工程」がリカーシブに起動され、いもずる式に、一つのファイルから、オブジェクトツリーの全ての情報を読み込む事ができる。

〈セクション11.12 プログラム構成例〉

セクション11の最初で示した図4の構成の計算機で、以上の工程を実行するプログラムの構成例を図14に示す。このプログラムは、オブジェクトを作成収容する機能、オブジェクトツリーが保持するオブジェクトの情報をオブジェクトの集合毎に異なるファイルに書き出す機能、これを読み込む機能、を実現する。

入力部401からの信号を「処理工程起動イベント受け付け工程」1401で分析し、対応した工程を起動する。また、各工程から処理工程起動イベントが出る場合もある。これも、同様に「処理工程起動イベント受け付け工程」1401で分析され、対応した工程を起動する。

起動される工程は、セクション11.1の「オブジェクトの作成収容工程」1402、セクション11.2の「オブジェクトへのポインタの情報を書き出す工程」1403、セクション11.3の「オブジェクトの情報を書き出す工程」1404、セクション11.4の「オブジェクトの集合を書き出す工程」1405、セクション11.5の「オブジェクトの情報を読み込む工程」1406、セクション11.6の「最初のオブジェクトを読み込む工程」1407、セクション11.7の「オブジェクトの集合を読み込む工程」1408、セクション11.8の「オブジェクト識別情報の組み合

わせからオブジェクトを特定する工程」1409がある。この他に、アプリケーション特有の工程を加えることが出来るが、本発明には関係ないので図14では省略している。

オブジェクトを作成しオブジェクトツリーに収容する場合は「オブジェクトの作成収容工程」1402を起動する。

オブジェクトツリー全体の情報を二次記憶部405の記録媒体のファイルに記録する場合は、オブジェクトツリーの根元のオブジェクト（図2ならばa0オブジェクト201）を指定して、「オブジェクトの集合を書き出す工程」1405を起動する。これから、「オブジェクトの情報を書き出す工程」1404が起動され、「オブジェクトへのポインタの情報を書き出す工程」1403が起動される。

必要なオブジェクトの情報をオブジェクトの集合毎に再現する場合には、まず「最初のオブジェクトを読み込む工程」1407を起動する。これにより、オブジェクトツリーの根元のオブジェクト（図2ならばa0オブジェクト201）が再現される。

オブジェクト識別情報の組み合わせをポインタに変換する場合に「オブジェクト識別情報の組み合わせからのオブジェクトを特定する工程」1409が起動される。これから「オブジェクトの集合を読み込む工程」1408が起動され、「オブジェクトの情報を読み込む工程」1407が起動され、必要なオブジェクトの集合の情報が読み込まれる。同時に再現されたオブジェクトのメモリアドレスが記録され報告される。

〈セクション12 オブジェクト管理装置〉

セクション11、12のプログラム構成例で示した、オブジェクト管理方法の実施例に相当する装置の実現例を図15に示す。

「入力部」1501からの入力を「入力分析手段」1510で分析し、対応する手段に信号を送る。信号を受ける可能性のある手段は、「オブジェクトの作成収容手段」1502、「オブジェクトへのポインタの情報を書き出す手段」1503、「オブジェクトの情報を書き出す手段」1504、「オブジェクトの集合を書き出す手段」1505、「オブジェクトの情報を読み込む手段」1506、「最初のオブジェクトを読み込む手段」1507、「オブジェクトの集合を読み込む手段」1508、「オブジェクト識別情報の組み合わせからオブジェクトを特定する手段」1509であり、それぞれセクション11、12で説明した、「オブジェクトの作成収容工程」1402、「オブジェクトへのポインタの情報を書き出す工程」1403、「オブジェクトの情報を書き出す工程」1404、「オ

「オブジェクトの集合を書き出す工程」1405、「オブジェクトの情報を読み込む工程」1406、「最初のオブジェクトを読み込む工程」1407、「オブジェクトの集合を読み込む工程」1408、「オブジェクト識別情報の組み合わせからオブジェクトを特定する工程」1409を、マイクロプロセッサなどのハードウェアで実現した手段である。この他に、アプリケーション特有の手段を加えることが出来るが、本発明には関係ないので図15では省略している。

「オブジェクトの作成収容手段」1502に信号が入力されると、「主記憶部」1511の情報を元にオブジェクトを作成し、「主記憶部」1511内のオブジェクトツリーに収容する。

「オブジェクトの集合を書き出す手段」1505への信号が、オブジェクトツリーの根元のオブジェクト（図2ならばa0オブジェクト201）を指定している場合は、「主記憶部」1511内部のオブジェクトツリー全体の情報を二次記憶部1512の記録媒体のファイルに記録する。なお、「オブジェクトの集合を書き出す手段」1505からの信号が「入力分析手段」1510を経由して「オブジェクトの情報を書き出す手段」1504に、さらに「オブジェクトの情報を書き出す手段」1504からの信号が「入力分析手段」1510を経由して「オブジェクトへのポインタの情報を書き出す手段」1503に入力され、オブジェクトツリー全体の情報を二次記憶部1512の記録媒体のファイルに記録する作業を補助する。

オブジェクトツリーの根元のオブジェクト（図2ならばa0オブジェクト201）を指定した信号が、「最初のオブジェクトを読み込む手段」1507へ入力されると、オブジェクトツリーの根元のオブジェクトが再生される。

オブジェクト識別情報の組み合わせを指定した信号が「オブジェクト識別情報の組み合わせからのオブジェクトを特定する手段」1509に入力されると、「入力分析手段」1510を経由して、「オブジェクトの集合を読み込む手段」1508、「オブジェクトの情報を読み込む手段」1507と連動して信号が入力され、必要なオブジェクトの集合の情報が読み込まれる。同時に再現されたオブジェクトのメモリアドレスが主記憶部1511に記録され、また信号として他の手段に伝えられる。

産業上の利用可能性

〈セクション13 産業上の利用可能性〉

以上のように、本発明のオブジェクト管理方法とその装置は、オブジェクト指向のプロプログラミングにおいて、オブジェクトとポインタを保存し再生する場合に広く用いる事が出来る。また、オブジェ

クト指向プログラミングで、複数の装置にオブジェクト情報を分散させる事や、簡易化されたハッシュメモリを実現するために使う事が出来る。さらに、オブジェクトの情報をエディターでファイル上に作成しこれをプログラムに読み込ませたり、通信回線から受信した情報から、オブジェクトやポインタを作成する使い方が出来る。これにより、インターネットで用いられているWebブラウザで高度な機能を実現する事が可能である。

請 求 の 範 囲

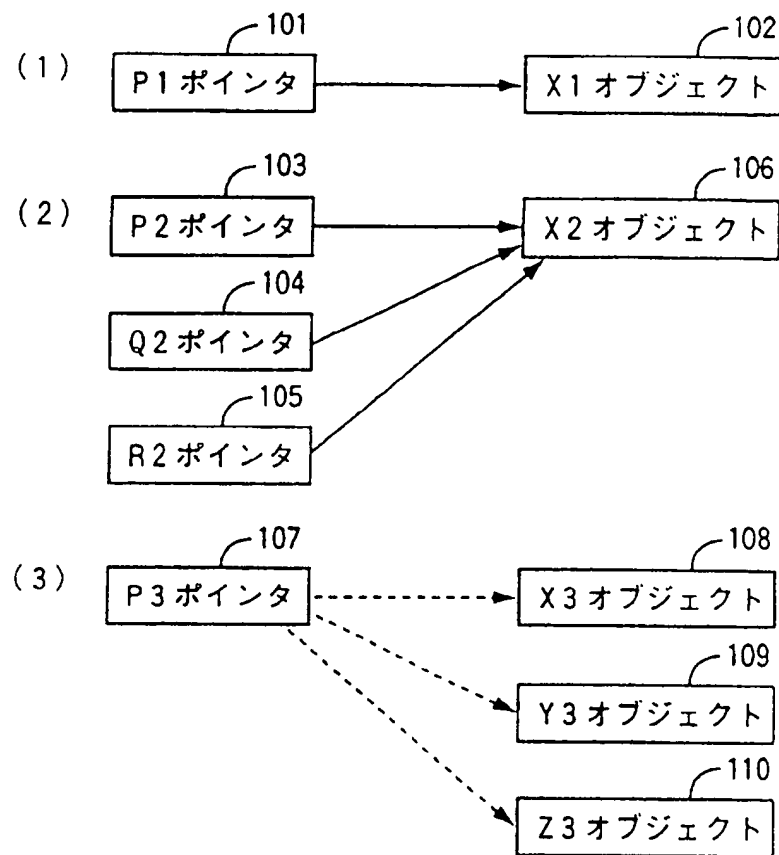
1. 「クラス識別情報を読み込みオブジェクトを作成する工程」を具備することを特徴とするオブジェクト管理方法。
2. 「オブジェクトの内容情報を読み込みオブジェクトに設定する工程」を具備することを特徴とする請求項1記載のオブジェクト管理方法。
3. (ア) オブジェクトのクラス識別情報を書き出す工程、(イ) オブジェクトの内容情報を書き出す工程、を具備することを特徴とする請求項2記載のオブジェクト管理方法。
4. (ア) オブジェクトの集合にその管理オブジェクトを対応させる工程、(イ) オブジェクト識別情報をオブジェクトに設定する工程、(ウ) オブジェクト識別情報の組み合わせからオブジェクトを特定する工程、を具備することを特徴とするオブジェクト管理方法。
5. 「管理オブジェクトを集合のメンバーとする工程」を具備することを特徴とする請求項4記載のオブジェクト管理方法。
6. 「ポインターをオブジェクト識別情報の連続に変換する工程」を具備することを特徴とする請求項5記載のオブジェクト管理方法。
7. (ア) オブジェクトのクラス識別情報を書き出す工程、(イ) オブジェクトの内容情報を書き出す工程、(ウ) クラス識別情報を読み込みオブジェクトを作成する工程、(エ) オブジェクトの内容情報を読み込みオブジェクトに設定する工程、を具備することを特徴とする請求項6記載のオブジェクト管理方法。
8. 「クラス識別情報を読み込みオブジェクトを作成する手段」を具備することを特徴とするオブジェクト管理装置。
9. 「オブジェクトの内容情報を読み込みオブジェクトに設定する手段」を具備することを特徴とする請求項8記載のオブジェクト管理装置。
10. (ア) オブジェクトのクラス識別情報を書き出す手段、(イ) オブジェクトの内容情報を書き出す手段、を具備することを特徴とする請求項9記載のオブジェクト管理装置。
11. (ア) オブジェクトの集合にその管理オブジェクトを対応させる手段、(イ) オブジェクト識別情報をオブジェクトに設定する手段、(ウ) オブジェクト識別情報の組み合わせからオブジェクトを特定する手段、を具備することを特徴とするオブジェクト管理装置。

12. 「管理オブジェクトを集合のメンバーとする手段」を具備することを特徴とする請求項11記載のオブジェクト管理装置。

13. 「ポインターをオブジェクト識別情報の連続に変換する手段」を具備することを特徴とする請求項12記載のオブジェクト管理装置。

14. (ア) オブジェクトのクラス識別情報を書き出す手段、(イ) オブジェクトの内容情報を書き出す手段、(ウ) クラス識別情報を読み込みオブジェクトを作成する手段、(エ) オブジェクトの内容情報を読み込みオブジェクトに設定する手段、を具備することを特徴とする請求項13記載のオブジェクト管理装置。

図 1



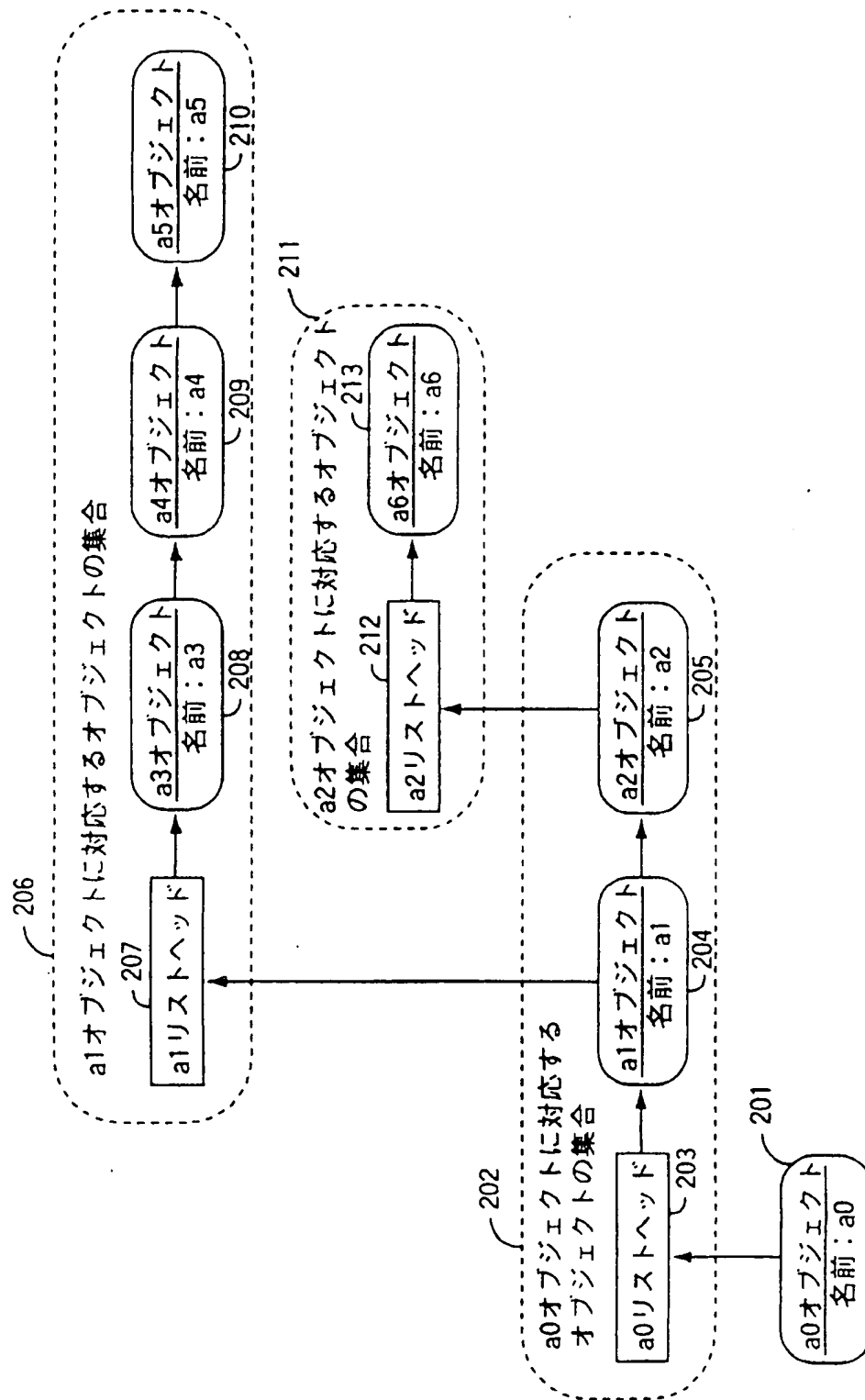


図 3

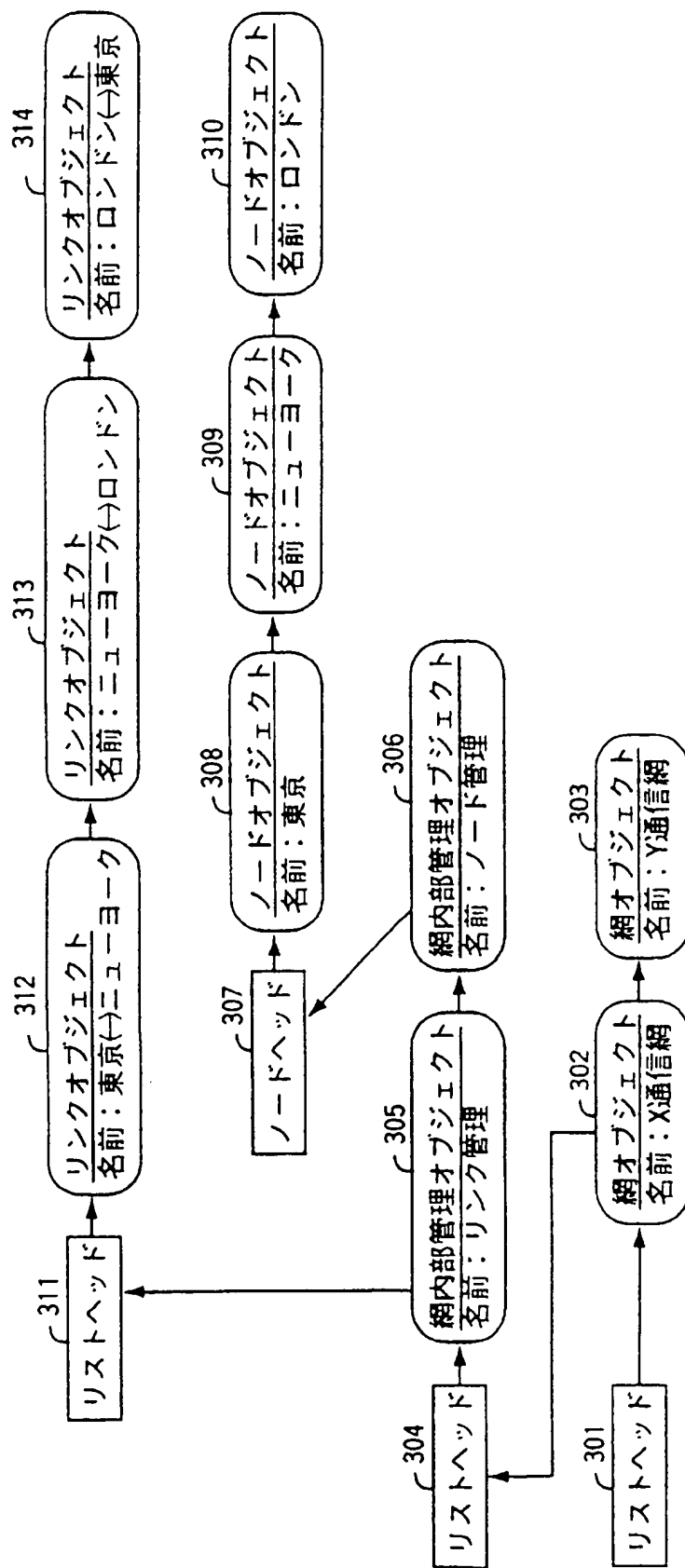
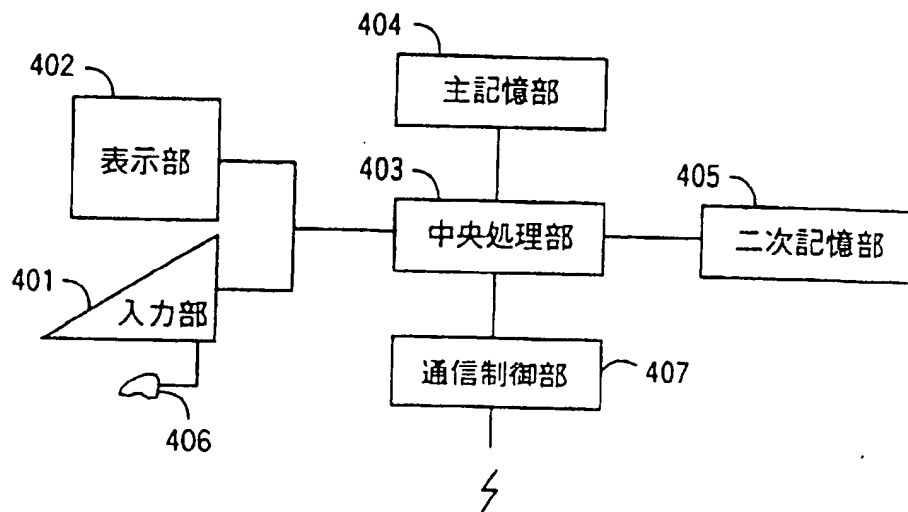


図 4



5 / 15

図 5

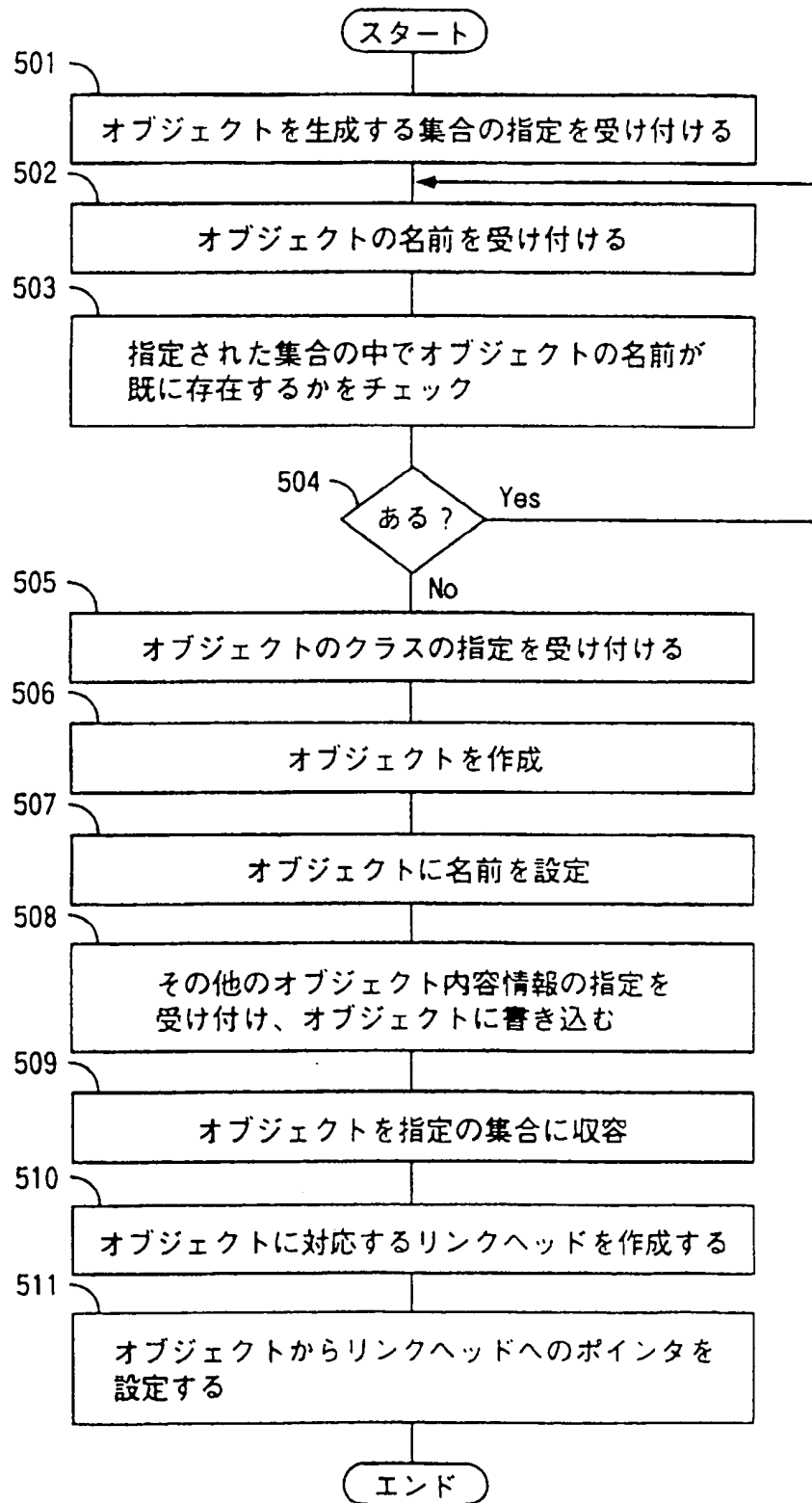


図 6

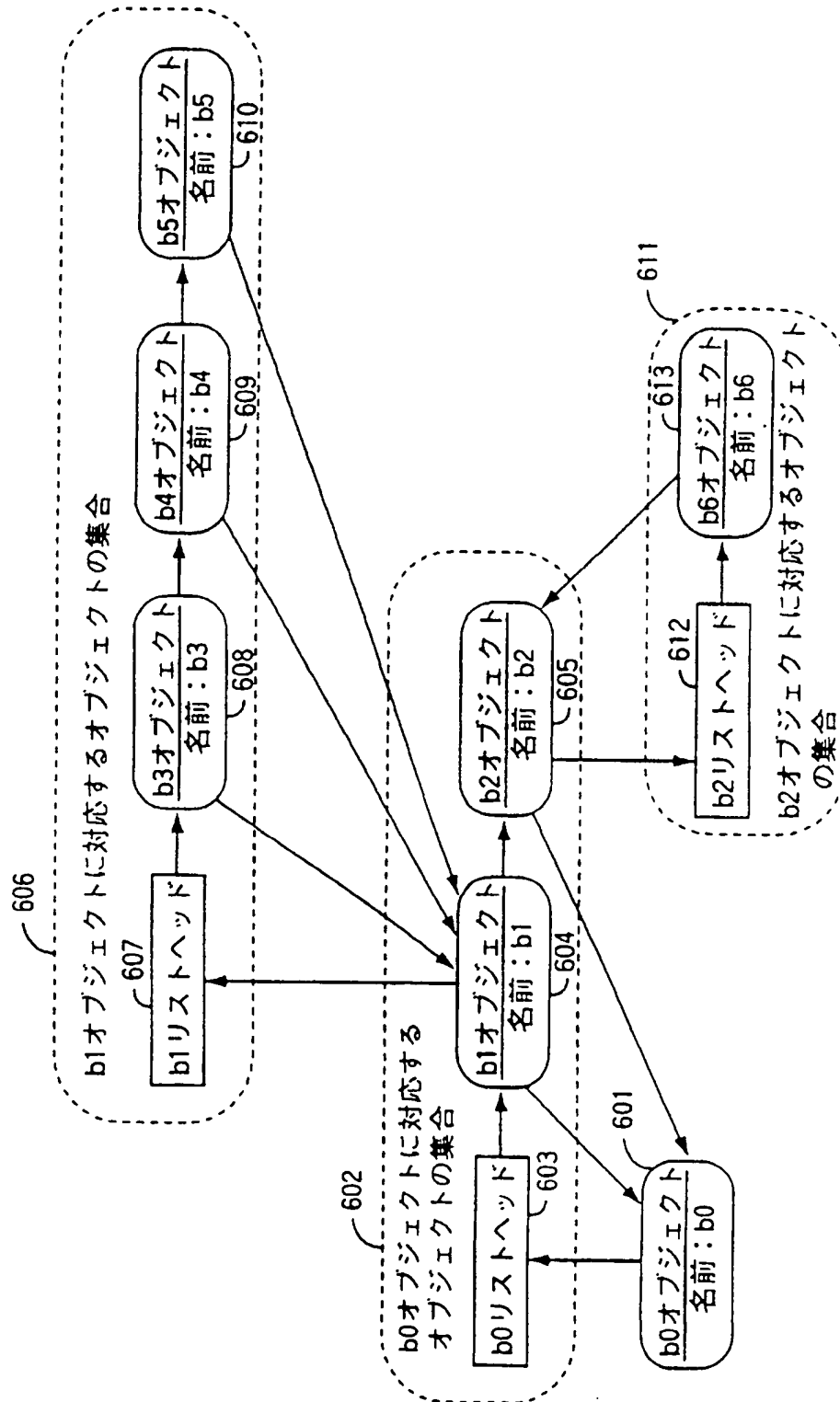


図 7

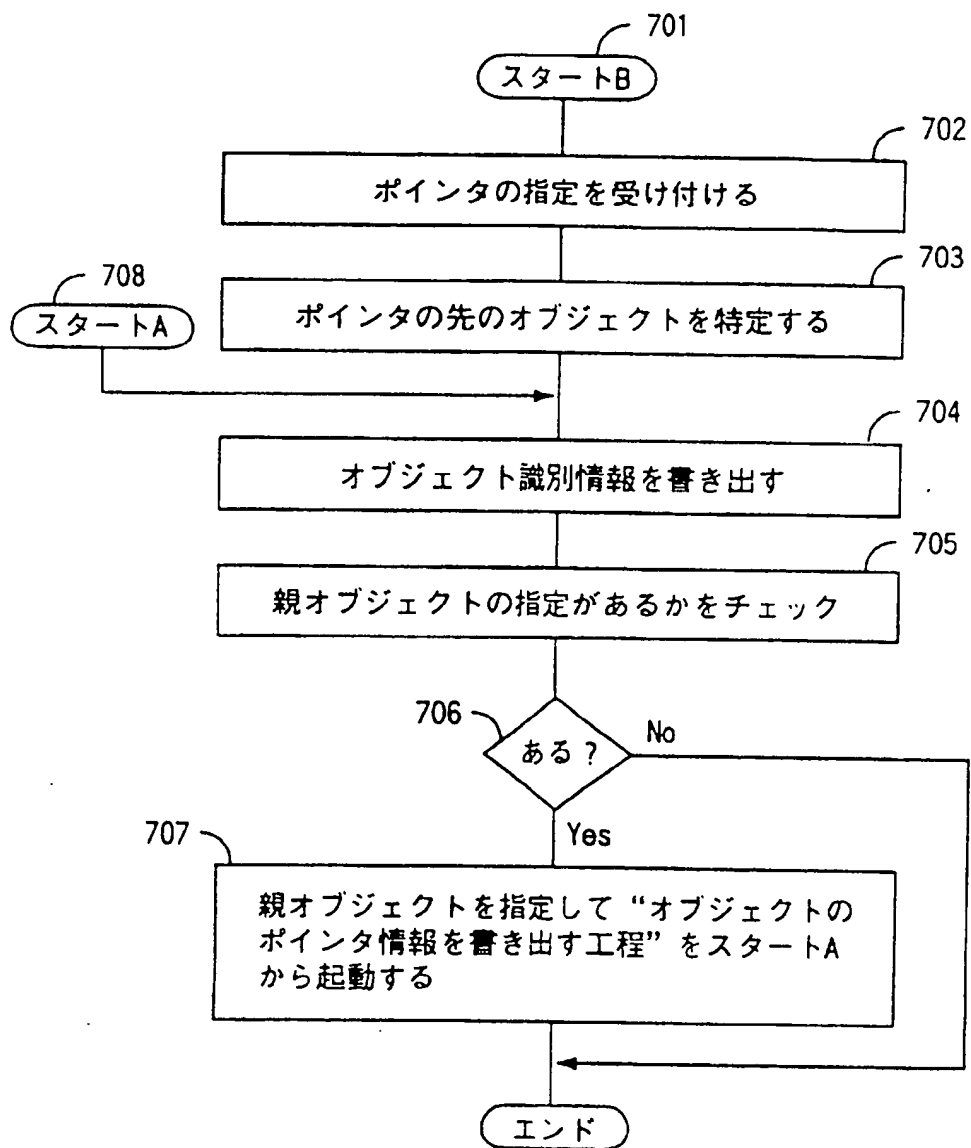


図 8

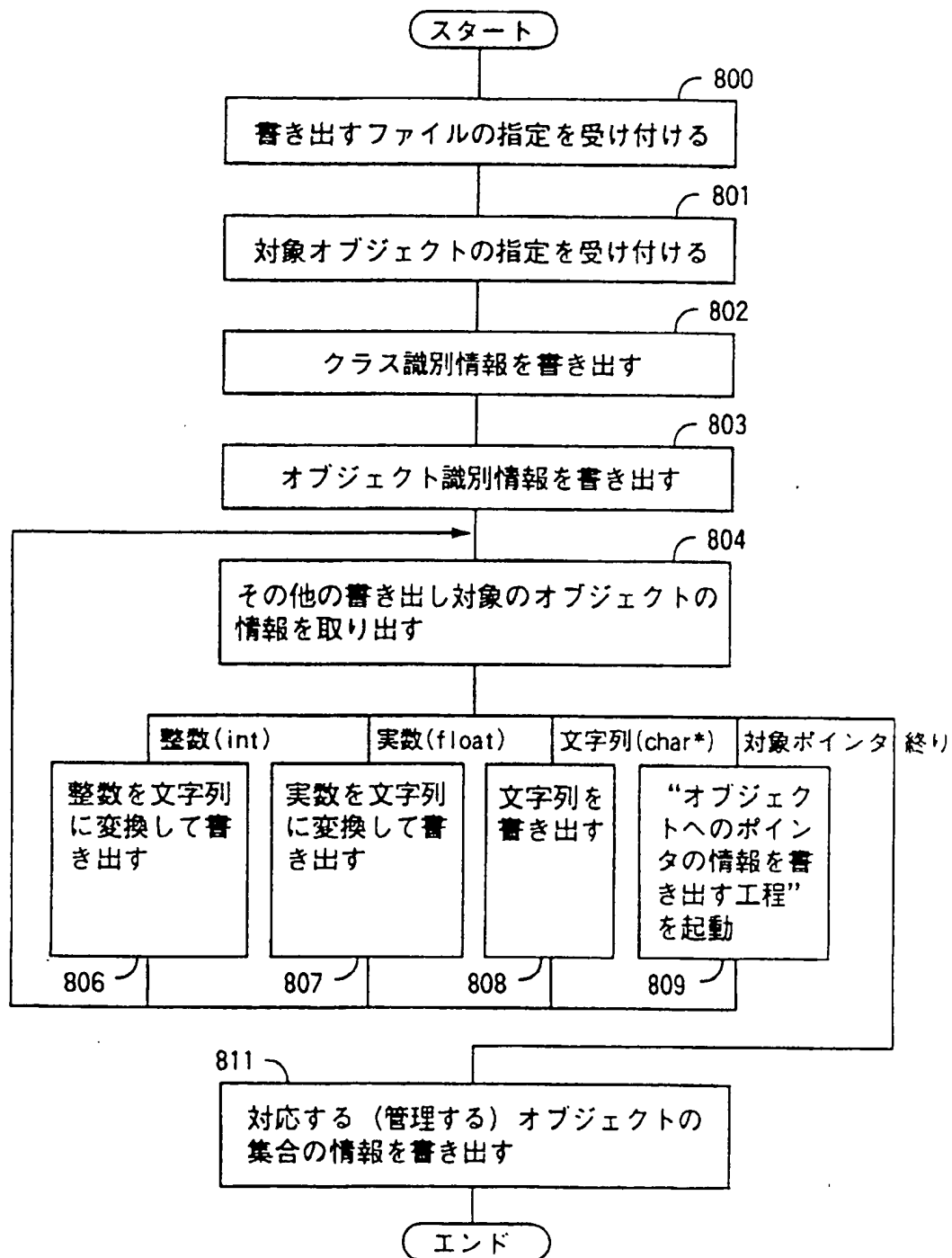
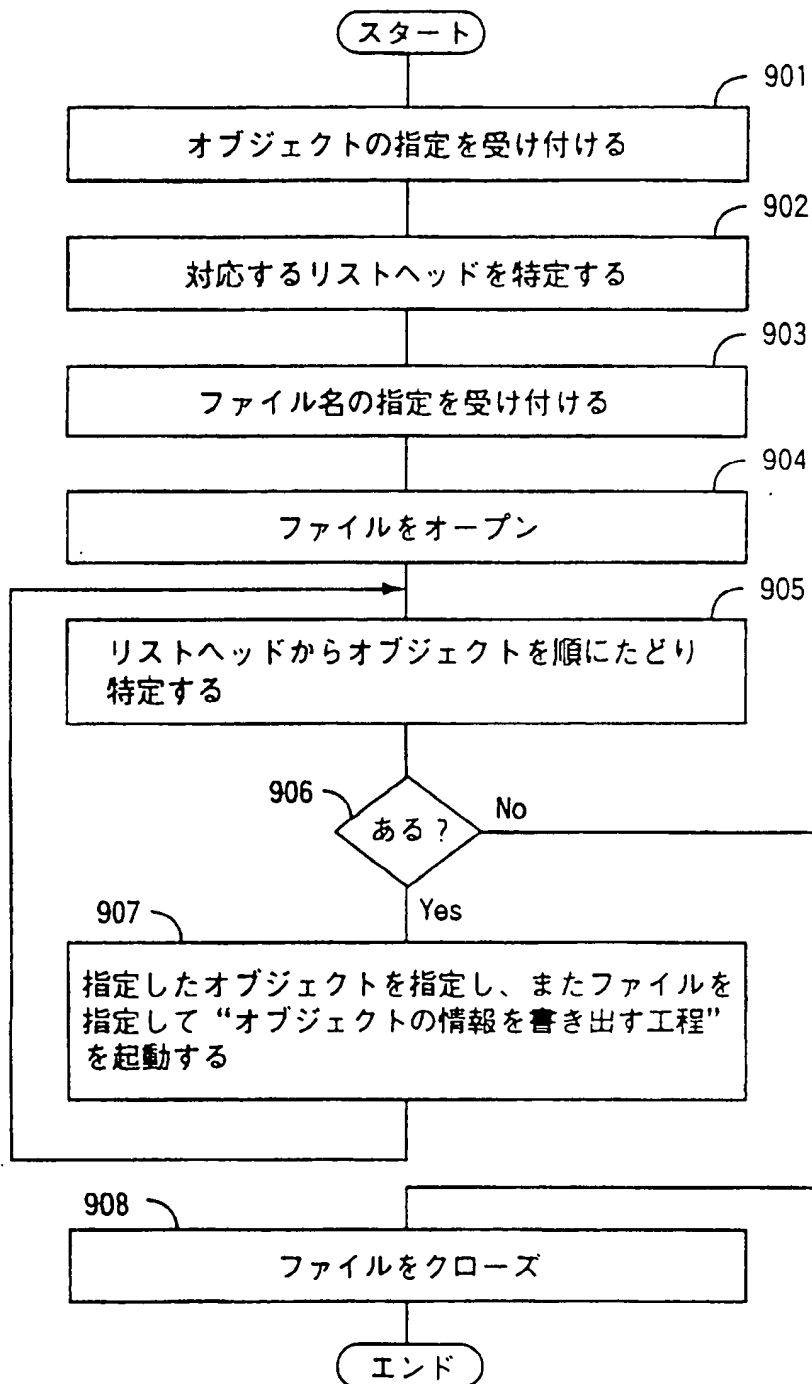


図 9



10/15

図10

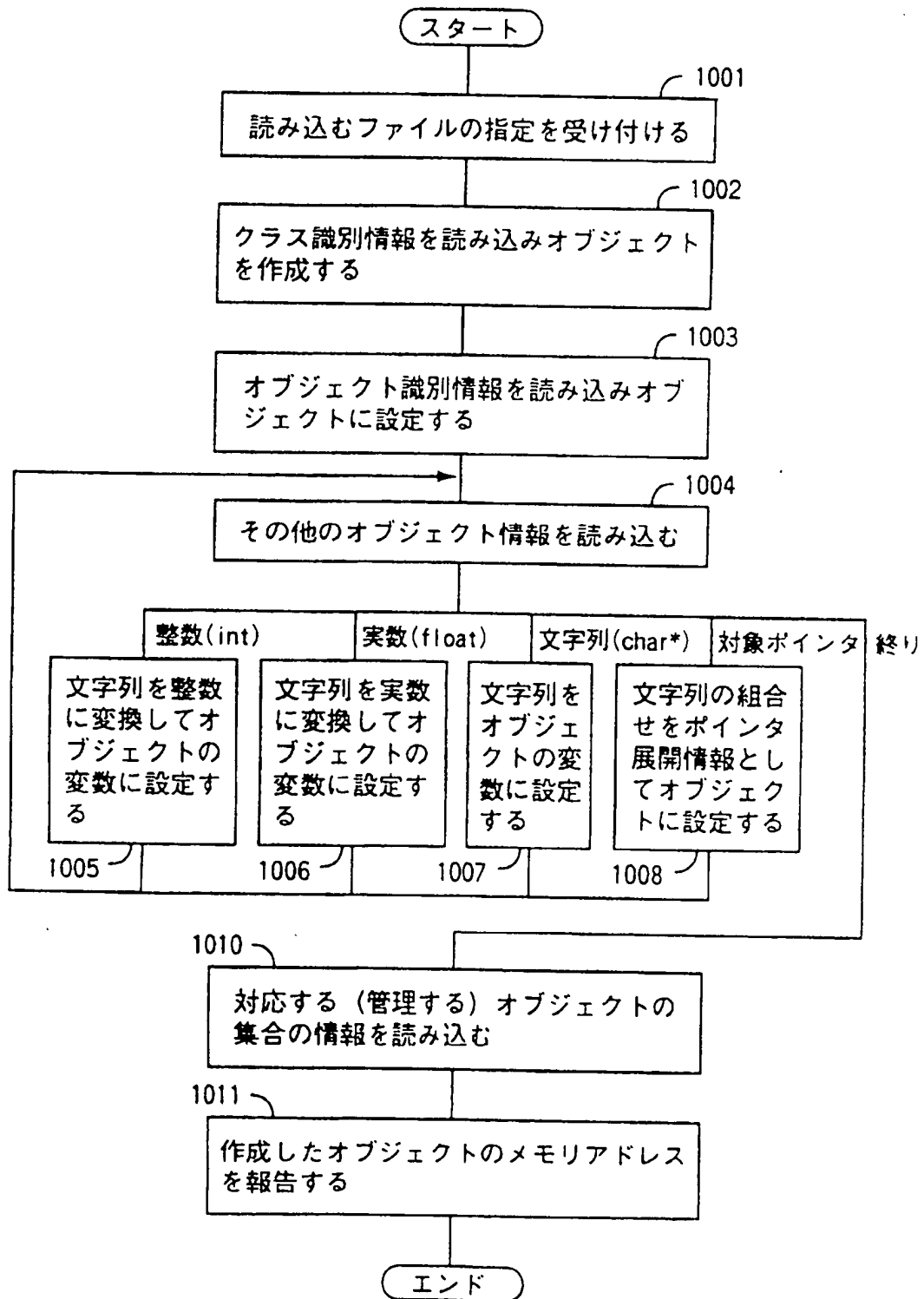


図 11

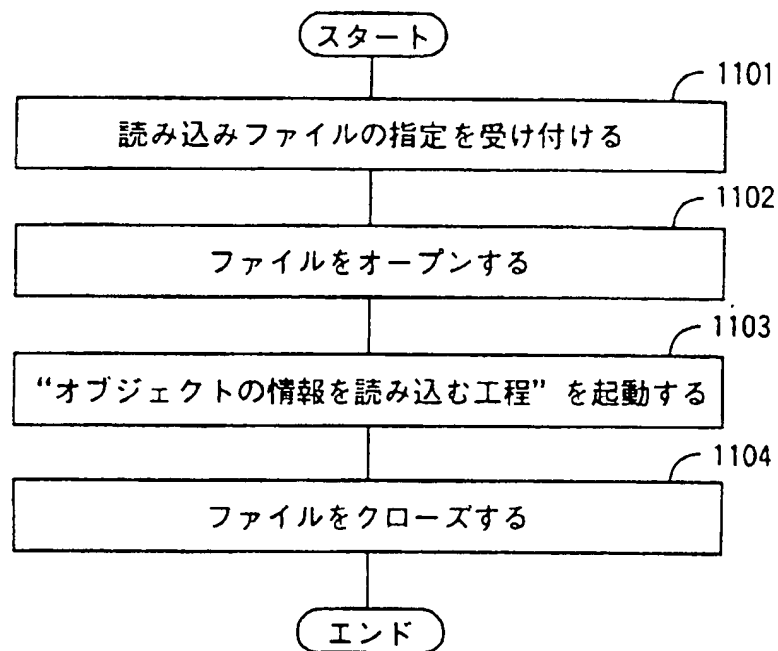
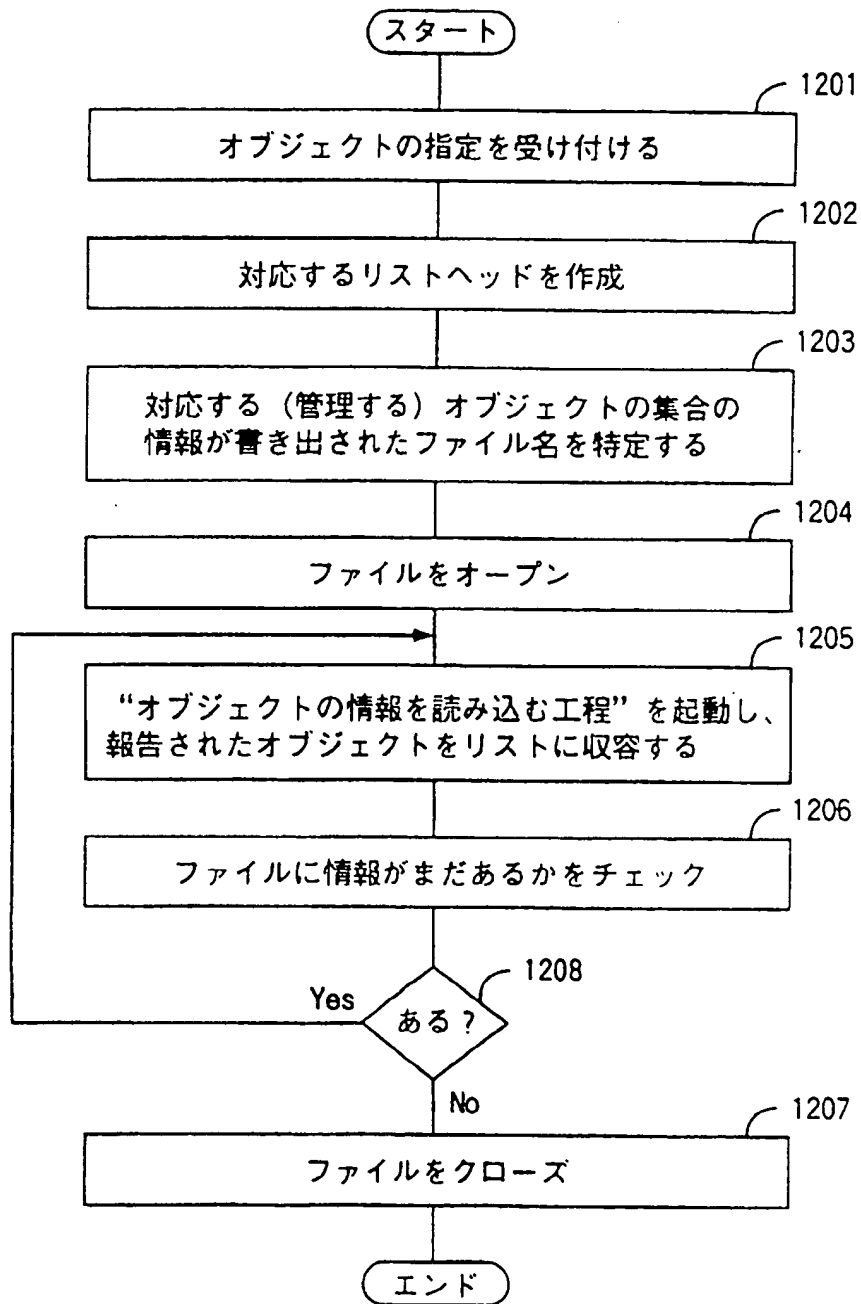


図 12



13/15

図13

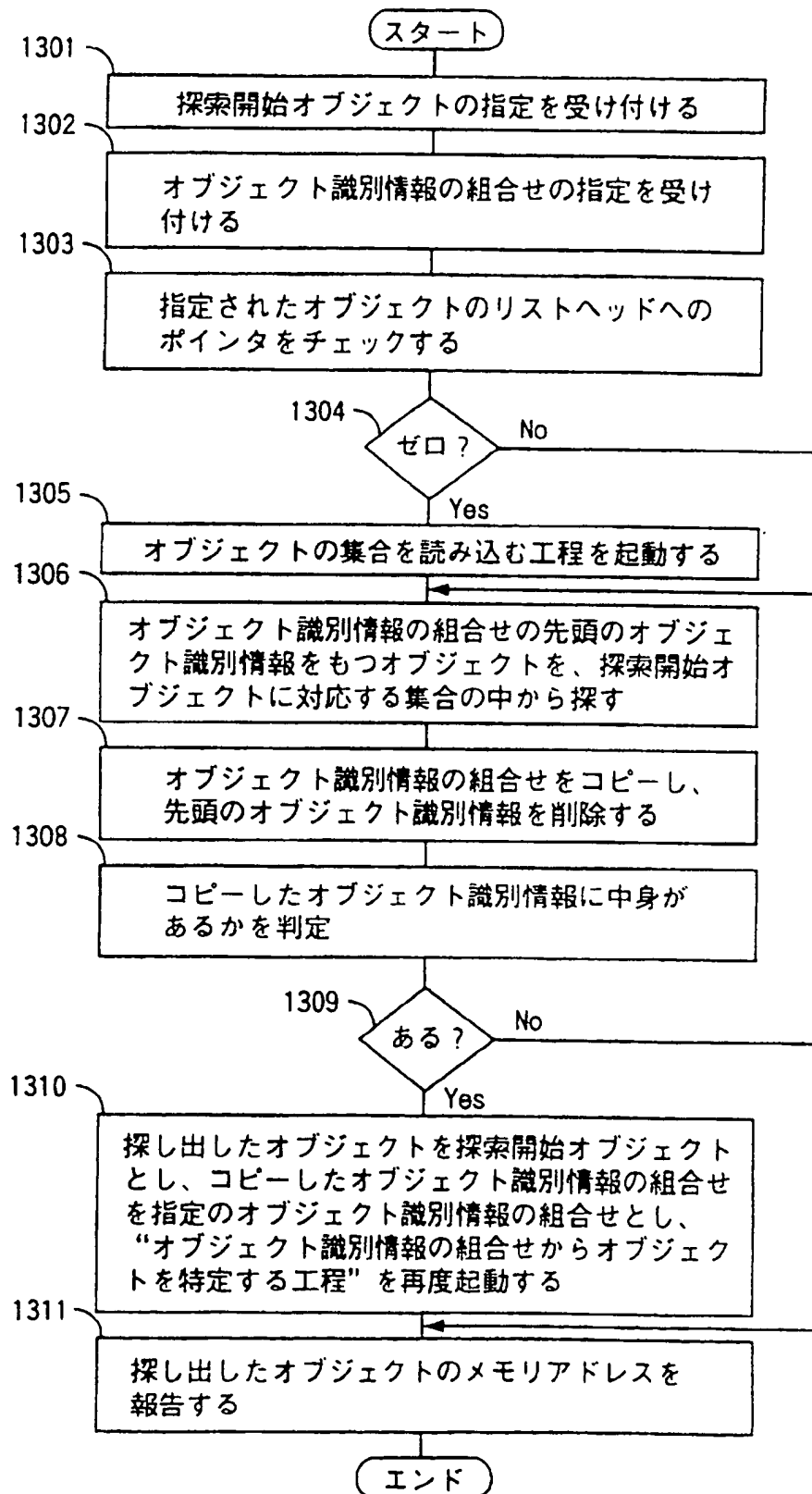


図 14

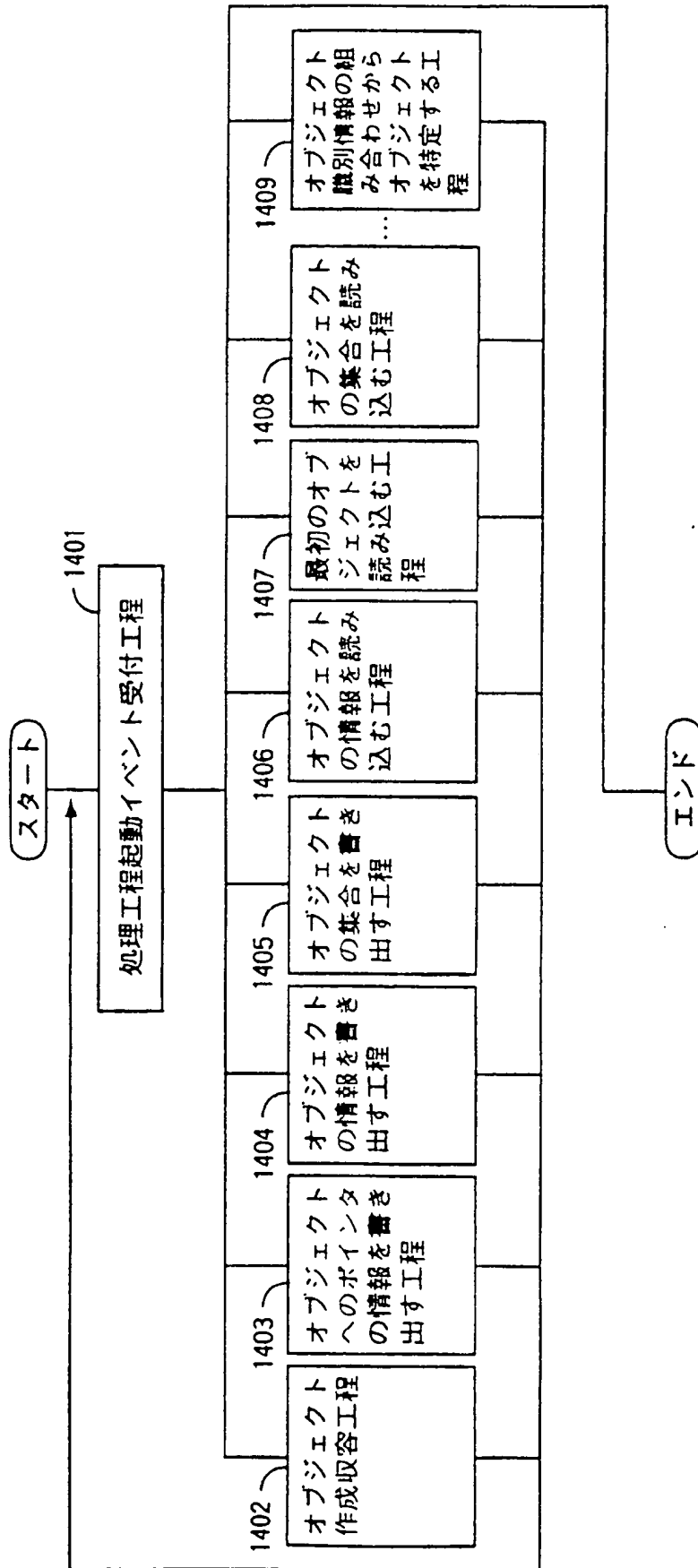
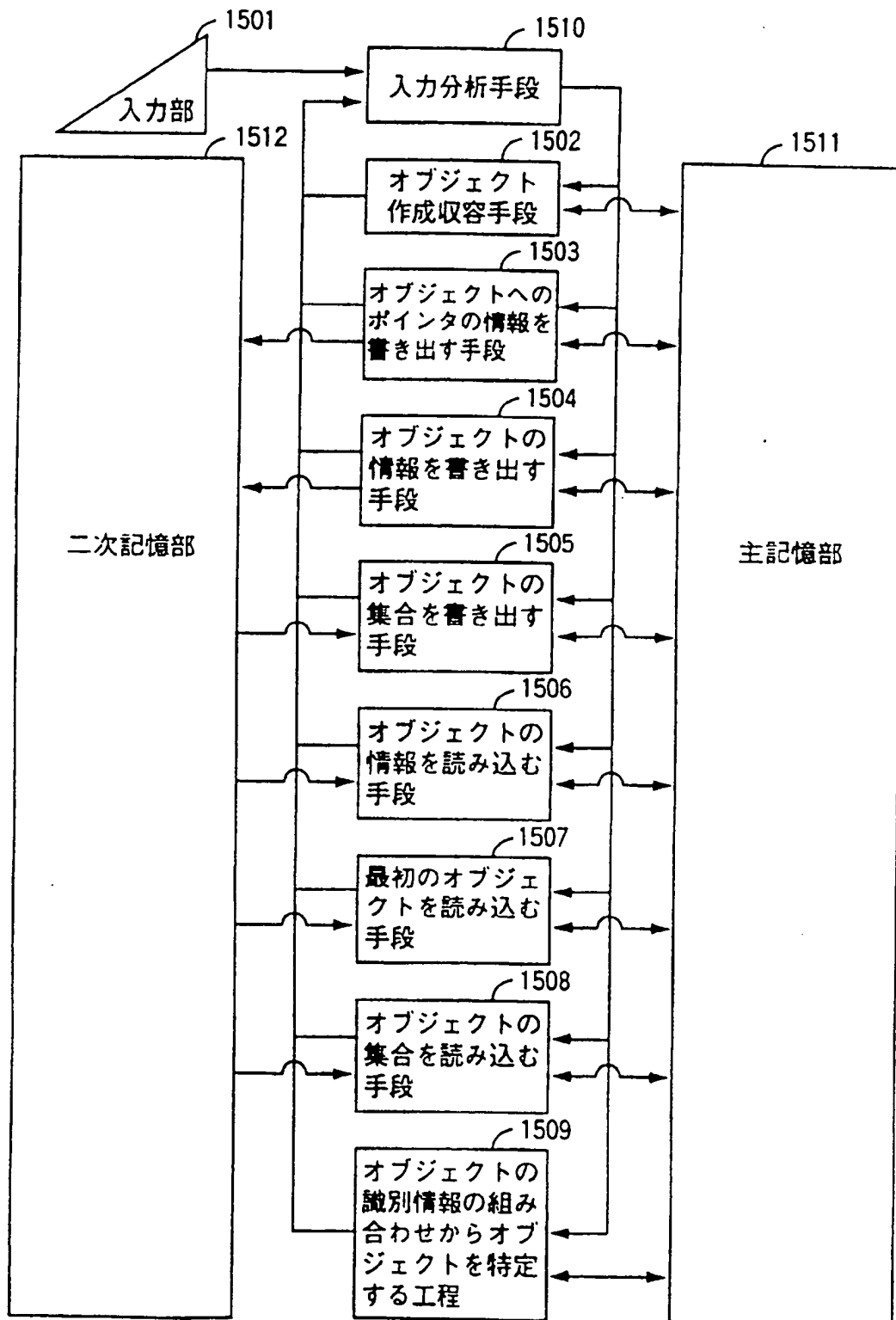


図 15



INTERNATIONAL SEARCH REPORT

International application No.

PCT/JP96/00227

A. CLASSIFICATION OF SUBJECT MATTER

Int. Cl⁶ G06F9/44

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

Int. Cl⁶ G06F9/44, G06F12/00

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Jitsuyo Shinan Koho	1979 - 1996
Kokai Jitsuyo Shinan Koho	1972 - 1994
Toroku Jitsuyo Shinan Koho	1994 - 1996

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	JP, 6-44128, A (Texas Instruments Inc.), February 6, 1994 (06. 02. 94) & US, 5297279, A & US, 5437027, A	1 - 14
A	JP, 5-225034, A (Fuji Xerox Co., Ltd.), September 5, 1993 (05. 09. 93) (Family: none)	1 - 14
A	JP, 7-141178, A (Fujitsu Ltd.), June 7, 1995 (07. 06. 95) (Family: none)	1 - 14
A	JP, 3-3039, A (Hitachi, Ltd., Hitachi Eng. Co., Ltd.), January 3, 1991 (03. 01. 91) (Family: none)	1 - 14

☐ Further documents are listed in the continuation of Box C.

☐ See patent family annex.

* Special categories of cited documents:

- "A" document defining the general state of the art which is not considered to be of particular relevance
- "E" earlier document but published on or after the international filing date
- "L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
- "O" document referring to an oral disclosure, use, exhibition or other means
- "P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search
April 26, 1996 (26. 04. 96)

Date of mailing of the international search report
May 14, 1996 (14. 05. 96)

Name and mailing address of the ISA/
Japanese Patent Office

Authorized officer

Facsimile No.

Telephone No.

A. 発明の属する分野の分類 (国際特許分類 (IPC))

Int. Cl¹ G06F9/44

B. 調査を行った分野

調査を行った最小限資料 (国際特許分類 (IPC))

Int. Cl¹ G06F9/44, G06F12/00

最小限資料以外の資料で調査を行った分野に含まれるもの

日本国実用新案公報 1979-1996年

日本国公開実用新案公報 1972-1994年

日本国登録実用新案公報 1994-1996年

国際調査で使用した電子データベース (データベースの名称、調査に使用した用語)

C. 関連すると認められる文献

引用文献の カテゴリー*	引用文献名 及び一部の箇所が関連するときは、その関連する箇所の表示	関連する 請求の範囲の番号
A	J P, 6-44128, A (テキサス インストルメンツ INC) 6. 2月. 1994 (6. 2. 94) & US, 5297279, A & US, 5437027, A	1-14
A	J P, 5-225034, A (富士ゼロックス株式会社) 5. 9月. 1993 (5. 9. 93) (ファミリーなし)	1-14
A	J P, 7-141178, A (富士通株式会社) 7. 6月. 1995 (7. 6. 95) (ファミリーなし)	1-14
A	J P, 3-3039, A (株式会社日立製作所, 日立エンジニアリング株式会社) 3. 1月. 1991 (3. 1. 91) (ファミリーなし)	1-14

☐ C欄の続きにも文献が列挙されている。☐ パテントファミリーに関する別紙を参照。

* 引用文献のカテゴリー

「A」 特に関連のある文献ではなく、一般的技術水準を示すもの

「E」 先行文献ではあるが、国際出願日以後に公表されたもの

「L」 優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献 (理由を付す)

「O」 口頭による開示、使用、展示等に言及する文献

「P」 国際出願日前で、かつ優先権の主張の基礎となる出願

の日の後に公表された文献

「T」 国際出願日又は優先日後に公表された文献であって出願と矛盾するものではなく、発明の原理又は理論の理解のために引用するもの

「X」 特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの

「Y」 特に関連のある文献であって、当該文献と他の1以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの

「&」 同一パテントファミリー文献

国際調査を完了した日

26. 04. 96

国際調査報告の発送日

14.05.96

国際調査機関の名称及びあて先

日本国特許庁 (ISA/JP)

郵便番号 100

東京都千代田区霞が関三丁目4番3号

特許庁審査官 (権限のある職員)

林 鋭

印

5B

9193

電話番号 03-3581-1101 内線 3545